Proyecto de Taller V:

Método RVR en la simulación de medidas de confiabilidad en redes

Informe Final
Febrero de 2000

Departamento de Investigación Operativa
Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Antonio Mauttone
Tutor: Dr. Ing. Héctor Cancela

Resumen

Una red está compuesta por un conjunto de nodos y un conjunto de aristas que comunican pares de nodos. La confiabilidad de una red es una medida que refleja la capacidad de la misma de continuar operativa frente a posibles fallos de algunos de sus componentes, y se define como la probabilidad de comunicación exitosa entre cierto conjunto de nodos de la red, dadas las probabilidades de funcionamiento de los componentes y la topología de la red. La evaluación exacta de esta medida es un problema NP-difícil, por lo que los algoritmos de cálculo exacto se hacen impracticables para redes de tamaño considerable. Una alternativa es utilizar métodos de simulación y en particular el método Monte Carlo. El algoritmo Monte Carlo estándar, directo o crudo requiere de un gran esfuerzo computacional para lograr estimaciones precisas en redes muy confiables. Por este motivo es de interés el estudio de algoritmos denominados *de reducción de varianza*. En este trabajo se estudia en particular la técnica de Reducción Recursiva de la Varianza aplicada al cálculo de confiabilidad en redes. Se realiza un estudio comparativo de tres algoritmos: el algoritmo exacto de Generación Completa de Estados y los algoritmos estimativos Monte Carlo Crudo y Reducción Recursiva de la Varianza. Se presentan los detalles de implementación de los algoritmos, los casos de prueba seleccionados para su testeo y los resultados numéricos obtenidos, así como las conclusiones extraídas a partir de los mismos. También se muestra la incorporación de las implementaciones realizadas a la herramienta HEIDI y la construcción de un sitio web para la difusión de este trabajo.

Palabras clave: confiabilidad en redes, grafos, simulación, método Monte Carlo, reducción de varianza, HEIDI

Contenido

Acerca de este informe

Este informe presenta las actividades realizadas, resultados obtenidos y conclusiones extraídas en el marco de este proyecto, durante el período que va de Marzo a Diciembre de 1999. El contenido del mismo es el siguiente:

1 - Introducción, donde se presenta el tema del proyecto y se describen a grandes rasgos las tareas realizadas.

2 - Descripción del software implementado para la representación del modelo, donde se detallan los siguientes pasos:
- Especificación de requerimientos.
- Especificación de análisis y diseño.
- Consideraciones de implementación.

3 - Algoritmos estudiados, donde se dan descripción y detalles de implementación para cada algoritmo estudiado.

4 - Plan de pruebas de validación y resultados obtenidos, donde se describen los casos de prueba utilizados para la validación de las implementaciones de los algoritmos.

5 - Plan de pruebas de comparación de la eficiencia de los algoritmos y resultados obtenidos, donde se describen los casos de prueba utilizados, así como los resultados y conclusiones.

6 - Integración con la herramienta HEIDI, donde se muestran los cambios realizados a dicha herramienta con el fin de incorporar nuestro trabajo a la misma.

7 - Difusión de resultados: Descripción del sitio web construido para la difusión del trabajo.

8 - Conclusiones y trabajos futuros.

9 - Referencias.

1 - Introducción

Este trabajo trata sobre el tema confiabilidad en redes. Las entidades relevantes en una red son nodos y conexiones entre nodos, y en general el principal objetivo buscado es lograr una comunicación segura entre nodos de la red. Las aplicaciones de modelos de redes se dan en los siguientes contextos:

- Redes telefónicas y de comunicación de datos.
- Redes de transporte.
- Arquitecturas de computadores.
- Redes de energía eléctrica.
- Sistemas de comando y control.

Los principales problemas a resolver en el análisis y el diseño de redes son, a grandes rasgos, los siguientes:

1 - Dado un conjunto de nodos que se desean comunicar entre sí, obtener una red óptima en algún sentido (por ejemplo, obtener la máxima cantidad posible de caminos distintos entre pares de nodos), sujeto a determinadas restricciones (por ejemplo, costo de conexión entre pares de nodos).

2 - Dada una red, evaluar de algún modo su confiabilidad (en el sentido de la comunicación entre nodos). Este tipo de problemas están fuertemente relacionados con problemas del tipo 1, donde en el proceso de búsqueda de la red óptima se deben comparar las confiabilidades de dos redes para escoger la mejor, o luego de obtener un resultado a partir de cierto procedimiento se debe evaluar su confiabilidad.

Nuestro trabajo se centra en la resolución de problemas del tipo 2.

Cabe aclarar que el modelo utilizado se adecua a realidades donde la información a comunicarse entre dos nodos se rutea de forma dinámica a través de la red. Esto significa que si bien puede existir un camino predeterminado para comunicar dos nodos, se puede escoger uno alternativo en caso de falla de algún componente intermedio del camino original (pensar en el ruteo de paquetes entre dos hosts de Internet). Se dejan de lado detalles como el control de congestión y de retrasos en la comunicación (problemas habituales en redes de datos y de transporte; ver análisis de performabilidad en [2]).

Al trabajar con nodos y conexiones entre nodos, para la construcción de un modelo formal se utilizan grafos (en lo que sigue se utilizarán las palabras red y grafo como sinónimos). En particular interesa definir una medida (o índice) de confiabilidad para una red, que esté basada en su topología y en la confiabilidad de sus componentes. La forma de obtener dichos índices varía dependiendo de los datos con que se cuente. Un dato imprescindible es el referente a la topología de la red, pero según se cuente o no con datos sobre las confiabilidades elementales de los componentes, se pueden adoptar dos enfoques (ver [1]):

- **Solidez o resistencia a fallos**: Es un índice calculado en base a parámetros usuales de grafos (cintura, cohesión, conectividad y otros). Se asume que todos los componentes son idénticos desde el punto de vista de sus eventuales fallos, por lo que este enfoque es útil cuando no se cuenta con información acerca de la confiabilidad de cada componente por separado (ver [21]).

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

El enfoque a tomar en este trabajo es el segundo, considerándose un modelo probabilístico basado en un grafo estocástico (se asignan probabilidades de funcionamiento a nodos y aristas).

Formalmente, el modelo puede especificarse de la siguiente forma:

G = (V, E, I)	grafo no dirigido
V	conjunto de nodos
E	conjunto de aristas
I	función de incidencia, $\boldsymbol{I : E \rightarrow V \times V}$ (pares no ordenados); asocia a una arista, sus dos nodos extremos
$\boldsymbol{r_V}$	probabilidad de funcionamiento de nodos, $\boldsymbol{r_V : V \rightarrow [0,1]}$
$\boldsymbol{r_E}$	probabilidad de funcionamiento de aristas, $\boldsymbol{r_E : E \rightarrow [0,1]}$

Existen básicamente tres medidas de interés sobre este tipo de redes:

- $\boldsymbol{R_v : G \rightarrow [0,1]}$ (confiabilidad global): Probabilidad de existencia de por lo menos un camino entre todo par de nodos del grafo.
- $\boldsymbol{R_{st} : G \times V \times V \rightarrow [0,1]}$ (confiabilidad fuente-terminal): Probabilidad de existencia de por lo menos un camino entre los nodos ***s*** y ***t***.
- $\boldsymbol{R_K : G \times K \subseteq V \rightarrow [0,1]}$ (confiabilidad entre terminales): Probabilidad de existencia de por lo menos un camino entre todo par de nodos del subconjunto ***K*** de nodos (denominados terminales).

En algunos casos se trabajará con las contrapartes de estas medidas desde el punto de vista de la anti-confiabilidad, definidas como:

- $\boldsymbol{Q_v = 1 - R_v}$
- $\boldsymbol{Q_{st} = 1 - R_{st}}$
- $\boldsymbol{Q_K = 1 - R_K}$

Por más detalles referentes al modelo y las medidas definidas sobre el mismo ver [2], [3] y [4].

Los cálculos exactos de estas medidas son problemas *NP-difíciles*, por lo tanto, no se conocen algoritmos de orden polinómico en la cantidad de componentes del grafo para resolverlos. Existen algoritmos exactos eficientes para tipos particulares de redes, pero en el caso general los métodos exactos se hacen impracticables por sus excesivos tiempos de ejecución, para redes relativamente grandes. Una alternativa es utilizar métodos de simulación. La opción más simple en términos de implementación es utilizar el método Monte Carlo estándar o crudo como forma de simular el comportamiento estocástico del sistema. En el contexto del cálculo de confiabilidad en redes, el método Monte Carlo crudo consiste en sortear un estado del grafo (equivalente a sortear el funcionamiento de sus nodos y aristas) y verificar su conexidad (o la condición que defina a la red como operativa o funcionando correctamente). Realizando varias replicaciones del mismo experimento, se obtiene una estimación de la medida requerida, como la frecuencia de aparición del suceso *"la red se encuentra en estado operativo"*. La desventaja importante de este método es la excesiva varianza de las estimaciones (que como consecuencia aumenta el tamaño del intervalo de confianza de la estimación), lo que hace que sus resultados sean poco precisos. En particular esto se da para redes de alta confiabilidad, donde se hace necesario realizar un número considerable de replicaciones para obtener estados de falla o no operativos. Esto motiva el estudio de métodos de reducción de la varianza para la simulación de medidas de confiabilidad. En particular, en este trabajo se estudia el método de reducción recursiva de la varianza (RVR, siglas en inglés de “recursive variance reduction”). El mismo surge como una alternativa dentro de la familia de algoritmos de reducción de varianza, donde para los ya conocidos (ver estado del arte en [19]), se observan las siguientes características:

- Utilización excesiva de memoria.
- Excesivo tiempo de ejecución (ya sea por la complejidad del algoritmo o por los tiempos de pre-procesamiento requerido en algunos).
- Buena performance solo para algunas clases de topologías y valores particulares de confiabilidades elementales de los componentes.

Nuestro trabajo surge con el objetivo de contar con una implementación de la técnica RVR aplicada al cálculo de confiabilidad en redes, y de poder evaluar la misma (en base a resultados numéricos) frente a otras técnicas. Concretamente se desean estudiar los tiempos de ejecución, varianzas y performance general con respecto a topologías particulares y valores particulares de confiabilidades de los componentes. Se requirió inicialmente de un relevamieto de información e interiorización en el tema, principalmente en base a artículos recientes. Luego se pasó a la implementación del modelo (se considera un modelo con fallas en aristas y nodos) y de los algoritmos a estudiar, en un determinado lenguaje de programación. Se realizaron pruebas en base a varias topologías con el fin de comparar las performances de los distintos algoritmos, buscando confirmar la efectividad de la reducción de varianza en redes confiables por parte del algoritmo RVR.

Este proyecto continúa la línea seguida en los siguientes Talleres V:

- G. Friss de Kereki, M. Maneyro, F. Robledo, A. Sabiguero (1996) *Modelos de Confiabilidad en Redes*
- F. Berruti, P. Pereyra, R. Cardozo (1997) *Modelos de Confiabilidad*
- G. Maquiel, P. Barrios (1996) *Interface para el modelado y diseño de redes de comunicaciones*

y los siguientes proyectos de investigación:

- Proyecto BID/Conicyt 153/92 *Modelado y simulación de redes de comunicaciones en ambientes inteligentes. Herramienta de diseño*
- Proyecto ECOS U93E03 *Programa de Cooperación Científica Francia-Uruguay*

donde se han estudiado diversas técnicas y algoritmos para el cálculo de confiabilidad en redes e incluso se ha estudiado la técnica RVR en un modelo de redes con fallas solamente en aristas.

Todos los trabajos antes mencionados (incluyendo el presente), se encuentran enmarcados en el plan de trabajo del grupo de "Modelado de Sistemas Complejos" del Departamento de Investigación Operativa del In.Co.

2 - Implementación del modelo

A continuación, se describe el software requerido para la representación del modelo propuesto para el testeo de los algoritmos que luego se estudiaron. El desarrollo de dicho software sigue los tres siguientes pasos:

- Requerimientos.
- Análisis y diseño.
- Implementación.

Especificación de requerimientos

A continuación se especifican los elementos que intervienen en el modelo probabilístico de una red (en notación formal) y luego las características que debe cumplir el software que lo implemente (en lenguaje natural).

Modelo matemático

Al modelo especificado en el capítulo 1 de este informe se agrega lo siguiente:

ϕ función de estructura, $\boldsymbol{\phi : G \rightarrow \{0,1\}}$; Dado un estado del grafo original (definido por los componentes del mismo que en ese instante funcionan), evalúa en 1 si la red se encuentra en estado operativo (esto es, si existe camino entre todo par de nodos, entre ***s*** y ***t***, o entre todo par de terminales del conjunto ***K***, dependiendo de la medida de interés) y en 0 en caso contrario.

Especificación en lenguaje natural

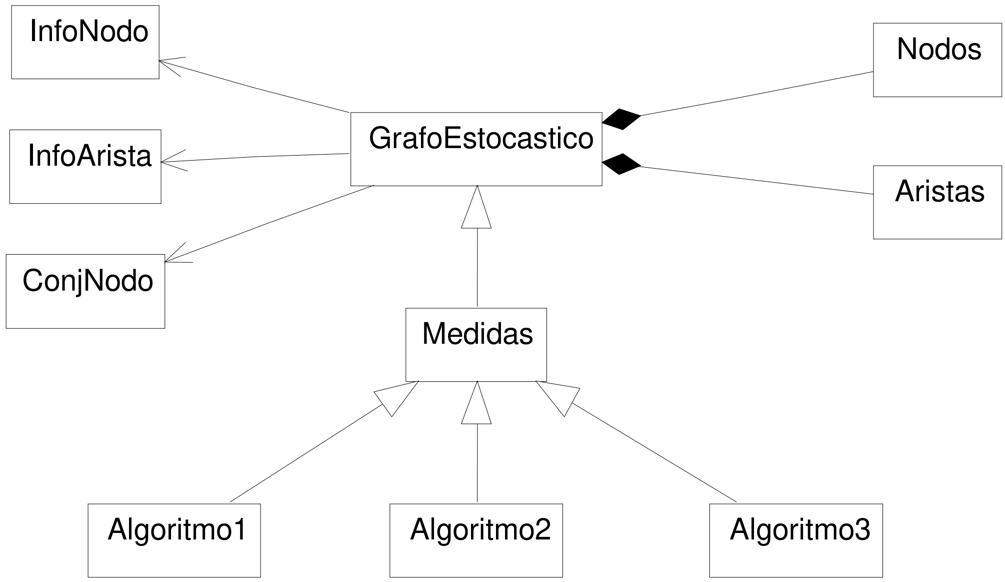
Para realizar pruebas sobre el modelo se requiere de la implementación de una biblioteca que soporte las siguientes características:

- Representación del modelo matemático grafo y en particular grafo estocástico (probabilidades asociadas a nodos y aristas).
- El grafo es no dirigido (las aristas pueden recorrerse en ambos sentidos).
- El grafo es simple (sin auto ni multi-aristas).
- Disponibilidad de operaciones para agregar y quitar nodos y aristas.
- Representación de nodos terminales (en particular interesa que la operación que informa si un nodo es terminal dado su identificador, sea de orden constante).
- Es deseable que la biblioteca sea extensible de forma de poder agregar nuevos algoritmos sin modificar los módulos originales.
- En relación con el punto anterior, se trata de no compartir información que no es común a distintos algoritmos (por ejemplo: marcas para distintos tipos de algoritmos).
- Posibilidad de almacenar información asociada a nodos y aristas, que depende del problema particular para el que en cierto momento se utilice la biblioteca.

Especificación de análisis y diseño

Para el análisis y diseño del software requerido se ha optado por una metodología orientada a objetos. Se utiliza para especificar tanto el análisis como el diseño, la notación **UML** [5].

Diagrama de clases



La *clase base* de la jerarquía permite representar grafos estocásticos (contiene la estructura de datos que lo representa y sus operaciones habituales).

En el *segundo nivel de la jerarquía* se cuenta con una clase (**Medidas**) que contiene los datos y operaciones necesarias para realizar los cálculos de las distintas medidas de confiabilidad sobre un grafo (entre ellas, la implementación de la función de estructura ϕ).

En el *tercer nivel de la jerarquía* se ubican las clases que implementan los algoritmos a estudiar. En cada una de las mismas se implementa un algoritmo particular, encapsulando las estructuras de datos y operaciones necesarias para el correcto funcionamiento del mismo.

A continuación se especifica el conjunto de operaciones para las clases **GrafoEstocastico** y **Medidas**.

Especificación de las operaciones para cada clase

Obs: Se asume que los nodos están identificados por números enteros en el rango 1..|*V*| y las aristas por pares de números enteros.

GrafoEstocastico (usa **InfoNodo**, **InfoArista**, **ConjNodo**, **Nodos**, **Aristas**)

crear();
// Crea un grafo estocástico vacío
cargar (arch : *in String*);
// Carga los datos del grafo desde un archivo
darCantNodos(): *Integer*;
// Devuelve la cantidad de nodos del grafo
darCantAristas(): *Integer*;
// Devuelve la cantidad de aristas del grafo
deshabilitarNodo(x : *in Integer*);
// Efectúa la baja lógica del nodo x

habilitarNodo(x : *in Integer*);
// Efectúa el alta lógica del nodo x
deshabilitarArista(x : *in Integer*; y : *in Integer*);
// Efectúa la baja lógica de la arista (x,y)
habilitarArista(x : *in Integer*; y : *in Integer*);
// Efectúa el alta lógica de la arista (x,y)
agregarArista(x : *in Integer*; y : *in Integer*);
// Efectúa el alta física de la arista (x,y)
darInfoNodo(x : *in Integer*): **InfoNodo**;
// Devuelve la información asociada al nodo x
darInfoArista(x : *in Integer*; y : *in Integer*): **InfoArista**;
// Devuelve la información asociada a la arista (x,y)
cambiarInfoNodo(x : *in Integer*; info : *in* **InfoNodo**);
// Modifica la información del nodo x
cambiarInfoArista(x : *in Integer*; y : *in Integer*; info : *in* **InfoArista**);
// Modifica la información de la arista (x,y)
darProbNodo(x : *in Integer*): *Real*;
// Devuelve la probabilidad de funcionamiento del nodo x
darProbArista(x : *in Integer*; y : *in Integer*): *Real*;
// Devuelve la probabilidad de funcionamiento de la arista (x,y)
cambiarProbNodo(x : *in Integer*; p : *in Real*);
// Modifica la probabilidad de funcionamiento del nodo x
cambiarProbArista(x : *in Integer*; y : *in Integer*; p : *in Real*);
// Modifica la probabilidad de funcionamiento de la arista (x,y)
marcarTerminal(x : *in Integer*);
// Marca al nodo x como terminal
marcarNoTerminal(x : *in Integer*);
// Marca al nodo x como no terminal
esTerminal(x : *in Integer*): *Boolean*;
// Indica si el nodo x es terminal
darAdyacentes(x : *in Integer*): **ConjNodo**;
// Devuelve un conjunto iterable de los nodos adyacentes a x (habilitados y
// alcanzables)
darTerminales(): **ConjNodo**;
// Devuelve un conjunto iterable de los nodos que son terminales
existeArista(x : *in Integer*, y : *in Integer*);
// Indica si existe arista entre el par de nodos (x,y)
habilitado(x : *in Integer*): *Boolean*;
// Indica si el nodo x está habilitado
habilitada(x : *in Integer*, y : *in Integer*): *Boolean*;
// Indica si la arista (x,y) está habilitada
darIesimaArista(i : *in Integer*, x : *out Integer*, y : *out Integer*);
// Devuelve la i_ésima arista del grafo (utilizada para iterar sobre el conjunto de aristas)
darGrado(x : *in Integer*) : *Integer*;
// Devuelve el grado del nodo x
habilitarTodo();
// Habilita todos los nodos y aristas del grafo

Medidas (hereda de **GrafoEstocastico**)

fi() : *Boolean*;
// Función de estructura. Indica si el estado del grafo es operativo o no (en el sentido de
// la existencia de caminos entre todo par de nodos del conjunto de terminales de
// interés)
setearST(s : *in Integer*, t : *in Integer*);
// Marca los nodos ***s*** y ***t*** del grafo como terminales (en este caso interesa la medida
// R_{st})
setearK(K : *in ConjNodo*);
// Marca los nodos del conjunto ***K*** como terminales (en este caso interesa la medida
// R_K)

setearV();
// Marca todos los nodos del grafo como terminales (en este caso interesa la medida
// $\boldsymbol{R_V}$)

Para cada algoritmo a estudiar se debe implementar una clase que herede de la clase **Medidas**, que encapsule las estructuras y las operaciones necesarias para el correcto funcionamiento del mismo.

Implementación

Para la implementación de los módulos especificados se ha optado por el lenguaje C++, principalmente por tres razones:

- Relativamente sencilla puesta en práctica del resultado de un diseño orientado a objetos.
- Manipulación a bajo nivel de las estructuras de datos (para lograr eficiencia en los algoritmos).
- Disponibilidad de un generador de números seudo-aleatorios de calidad aceptable.
- Compatibilidad con trabajos anteriores (principalmente pensando en la integración con HEIDI).

Diseño de las estructuras de datos

A continuación se especifica el diseño de las estructuras de datos y se justifican las decisiones tomadas respecto a las mismas.

Para la mayoría de las estructuras se utilizan arreglos dinámicos (como convención se denomina estructura dinámica a aquella para la cual se determina su tamaño en tiempo de ejecución; esto no significa que tenga la capacidad de redimensionamiento).

Nodos

La información de nodos se representa mediante un arreglo dinámico de largo n ($n = |V|$), donde cada elemento del arreglo contiene cuatro campos:

- **habilitado** (*boolean*): Indica si el nodo está en funcionamiento.
- **r** (*float*): Probabilidad de funcionamiento.
- **info** (**InfoNodo**): Información del nodo.
- **term** (*boolean*): Indica si el nodo es terminal.

La elección del arreglo se basa en la eficiencia requerida por cualquier operación que obtiene información de un nodo a partir de su identificador (que es el lugar que ocupa en el arreglo más uno, pues los arreglos en C/C++ comienzan en la posición 0). La estructura es dinámica pues su tamaño se determina al momento de cargar el grafo desde un archivo. Dicha estructura está implementada en la clase **Nodos**.

Aristas

Para la estructura de adyacencias la elección inmediata es una matriz de tamaño $n \times n$. Dado que el grafo es no dirigido se puede utilizar solo la mitad de la misma (triangular superior), y teniendo en cuenta que es simple (no tiene auto aristas), se puede prescindir de la diagonal. Teniendo en cuenta lo anterior se ha utilizado una matriz triangular de tamaño $n \times (n - 1) / 2$.

Dado que el lenguaje no provee las facilidades para la manipulación de este tipo de matrices, se utiliza un arreglo dinámico de una dimensión con un mapeo de pares (i, j) que representan aristas, en posiciones del arreglo. En la posición (i, j) de la matriz, se almacena un puntero, que de existir tal arista, apuntará a una estructura con los siguientes datos:

- **habilitada** (*boolean*): Indica si la arista está en funcionamiento.
- **r** (*float*): Probabilidad de funcionamiento.
- **info** (**InfoArista**): Información de la arista.

La elección de la matriz por sobre la lista (si bien los grafos no serán densos) se basa en la eficiencia requerida por la operación que obtiene información de una arista a partir de los identificadores de sus dos nodos. La estructura es dinámica por la misma razón que para los nodos.

Conjunto de nodos y aristas

Los conjuntos de (identificadores de) nodos y aristas se implementan mediante un arreglo dinámico encapsulado en una clase en la que se incluye un iterador sobre el conjunto. Solamente se almacenan los identificadores (números enteros en el rango 1..*n* para nodos y en el caso de aristas, pares de enteros).

Especificación de la biblioteca

Se detallan a continuación los dos puntos principales a tener en cuenta por parte del usuario de la biblioteca:

1 - Clases que la componen.
2 - Procedimiento de extensión.

Clases

- **GrafoEstocastico**: Implementa un grafo estocástico y contiene todas las operaciones para su manipulación.
- **ConjNodo**: Representa un conjunto iterable de identificadores de nodos.
- **InfoNodo**: Clase genérica para la información de nodos. En problemas particulares, se debe modificar, agregándose los datos y operaciones particulares.
- **InfoArista**: Análoga a la clase **InfoNodo**.
- **Medidas**: Contiene las estructuras y operaciones necesarias para la evaluación de las funciones de estructura para cada caso ($\boldsymbol{R_{st}}$, $\boldsymbol{R_K}$ o $\boldsymbol{R_V}$).

Extensión

- Para implementar un nuevo algoritmo se debe crear una nueva clase que herede de la clase **Medidas** y que además contenga la implementación del algoritmo particular.
- Si es necesario contar con información particular para nodos y aristas (por ejemplo: capacidades y/o costos) se deben agregar en las clases **InfoNodo** e **InfoArista** todos los datos y operaciones necesarias para el problema particular a resolver.

3 - Algoritmos estudiados

En este capítulo se presentan los algoritmos estudiados y los detalles de implementación para cada uno. Los mismos se pueden dividir en dos categorías:

- **Exactos**: Realizan una evaluación exacta de la medida requerida.
- **Estimativos**: Realizan una estimación de la medida en base a métodos de simulación.

Dentro de la primer categoría se considera el algoritmo de **Generación Completa de Estados** (en adelante **GCE**).

Para la segunda, se considera en una primera instancia un algoritmo de simulación de tipo **Monte Carlo estándar** o "crudo" (**MCC**) y luego una variante más elaborada denominada **Reducción Recursiva de la Varianza** (**RVR**).

Los tres algoritmos se implementaron en clases separadas, cada una de las cuales hereda de la clase **Medidas** (a los efectos de poder evaluar la función de estructura para un estado cualquiera del grafo).

A continuación se presentan los detalles de implementación de la función de estructura y más adelante se describen las ideas de cada uno de los algoritmos junto con sus detalles de implementación.

Función de estructura

Todos los algoritmos implementados requieren de la evaluación de la función de estructura, definida como:

$$\phi : G \rightarrow \{0,1\}$$

que vale 1 si la red se encuentra en estado operativo y 0 en caso contrario.

En nuestro caso particular, el estado de la red depende del estado de sus componentes (nodos y aristas que pueden estar o no en funcionamiento), y se considera un estado como operativo cuando se cumple la condición de conexidad (existencia de por lo menos un camino entre todo par de nodos, entre ***s*** y ***t***, o entre todos par de terminales del conjunto ***K***, dependiendo de la medida de interés).

Formalmente, se puede definir un **Sistema Binario Estocástico** (**SBE**) como aquel que falla aleatoriamente, en función de la falla aleatoria de sus componentes. En el caso de los grafos estocásticos el sistema ***G*** está formado por sus componentes ($c \in V \cup E$ -nodos y aristas-) que pueden fallar. Para todo estado ***G'*** del sistema original ***G*** tal que $G' \subseteq G$ (grafo obtenido eliminando del original los componentes que en un momento dado se encuentran en estado no operativo) la función de estructura evalúa de la siguiente forma:

$\phi(G') = 1$	si los componentes de ***G'*** funcionan, los de ***G – G'*** fallan y el sistema funciona
$\phi(G') = 0$	si los componentes de ***G'*** funcionan, los de ***G – G'*** fallan y el sistema falla

Es necesario establecer un criterio para definir la confiabilidad en una red cuando alguno de los terminales no funciona. Se ha optado por la definición *coherente* de confiabilidad, para la cual en un estado donde por lo menos un terminal se encuentre no operativo, la función de estructura evalúa siempre en 0.

Un sistema de tipo **SBE** se dice *coherente* cuando cumple con las siguientes propiedades:

$\phi(\boldsymbol{G}) = 1$
$\phi(\emptyset) = 0$
$\phi(\boldsymbol{G'}) \leq \phi(\boldsymbol{G}) \quad \forall\ \boldsymbol{G'} \subset \boldsymbol{G}$

La primer propiedad establece que un grafo donde todos sus componentes funcionan está en estado operativo, mientras la segunda establece que cuando ningun componente funciona, el grafo no está en estado operativo.

La tercer propiedad (monotonía) asegura que un grafo en estado no operativo no puede pasar a un estado operativo si funcionan menos componentes. Teniendo en cuenta esta última propiedad, una definición *no coherente* para la confiabilidad establecería un estado como operativo cuando existe por lo menos un camino entre todo par de terminales que en ese momento se encuentran operativos. Observar que en el modelo de fallas solo en aristas esta última definición no tiene sentido. Por más detalles referentes a sistemas coherentes ver [6] y [2].

En nuestro caso, un estado operativo está ligado a la existencia de por lo menos un camino entre todo par de nodos terminales, por lo que puede verificarse mediante una recorrida en profundidad partiendo desde uno de los mismos. Considerando que un estado es no operativo también cuando algún terminal no funciona, este chequeo debe incluirse en la implementación antes de comenzar la recorrida.

La descripción del algoritmo para la evaluación de la función de estructura es la siguiente:

```
function fi(G') : {0,1}
      foreach v ∈ K do
            if (v no operativo) then
                  return 0;
      end for;
      alcanzados := 0;
      v := terminalArbitrario();
      DFS (v);
      if (alcanzados = |K |) then
            return 1;
      else
            return 0;
end fi;



procedure DFS (v)
      marcar v como visitado;
      if (v es terminal) then
            alcanzados++;
      foreach u ∈ adyacentes(v)
            if (u no visitado) then
                  DFS (u);
      end for;
end DFS;
```

Función de estructura

Algoritmo de Generación Completa de Estados

Este algoritmo realiza el cálculo exacto de la medida de confiabilidad generando todos los $2^{|V|+|E|}$ estados posibles del grafo y evaluando cuales son operativos. Para aquellos que lo son se acumula su probabilidad de ocurrencia (calculada como el producto de las probabilidades de los componentes que funcionan y los complementos de las probabilidades de los que no funcionan). En otras palabras, se acumulan las probabilidades de todos los estados que suman en la probabilidad del suceso *"la red se encuentra en estado operativo"*. Para su implementación se utiliza la técnica de Backtracking.

```
procedure GCE(G, i)
      if  i < (|V|+|E|) then
            i++;
            deshabilitarComponente(i);
            GCE(G, i);
            habilitarComponente(i);
            GCE(G, i);
      else
            if fi(G) then
                  R = R + probSubGrafo(G);
end GCE;



function probSubGrafo(G)
      result:=1;
      foreach c ∈ V ∪ E do
            if (c  operativo) then
                  result:=result * rc;
            else
                  result:=result * (1-rc);
      end for;
      return result;
end probSubGrafo;
```

Generación Completa de Estados

La complejidad del algoritmo es $O(2^{|V|+|E|})$ ya que se deben generar todos los estados posibles del grafo.

Algoritmo Monte Carlo Crudo

Como se definió anteriormente, dado un grafo ***G***, un estado o subgrafo ***G'*** de ***G*** en un instante particular queda definido por los componentes originales de ***G*** que en ese instante funcionan. La idea de este algoritmo se basa en obtener una estimación del valor de la confiabilidad de la red mediante el sorteo de ***N*** estados ***G'$_i$*** del grafo original ***G***. En definitiva, la confiabilidad de la red se calcula como la frecuencia de ocurrencia del suceso *"todos los nodos terminales funcionan y están conectados entre sí"*. Formalmente se puede decir que se estima la medida original ***R*** mediante el estimador ***R$_{est}$*** definido a partir de la muestra ***G'$_1$***, ***G'$_2$***, …, ***G'$_N$*** de la siguiente forma:

$$R_{est} = \frac{1}{N}\sum_{i=1}^{N} 1_{\{G_i^{'}\ \text{operativo}\}}$$

Notación: $\mathbf{1}_A$ es la función indicatriz del conjunto (evento) A. Se define como $\mathbf{1}_A(\omega) = 1$ si $\omega \in A$ y $\mathbf{1}_A(\omega) = 0$ si $\omega \notin A$, o de otra forma $\mathbf{1}_A(\text{true}) = 1$ y $\mathbf{1}_A(\text{false}) = 0$.

La calidad de la estimación se evalúa mediante el cálculo de intervalos de confianza cuya amplitud es proporcional a la varianza del estimador (ver observación 1 del Apéndice 1). La expresión para dicha varianza es la siguiente:

$$Var(R_{est}) = \frac{R(1-R)}{N}$$

donde el numerador corresponde a la varianza de una variable aleatoria con distribución de Bernoulli. Al no disponerse del valor de ***R*** se construye un estimador para Var(***R$_{est}$***) de la siguiente forma:

$$Var(R_{est}) = \frac{1}{N(N-1)}\sum_{i=1}^{N}\left(1_{\{G_i^{'}\ \text{operativo}\}} - R_{est}\right)^2$$

En el caso particular de las variables aleatorias de Bernoulli, la expresión anterior se simplifica a:

$$Var(R_{est}) = \frac{R_{est}(1-R_{est})}{N-1}$$

El algoritmo implementado recibe el grafo, un identificador de semilla para el generador de números seudo-aleatorios y la cantidad de replicaciones ***N***. Se utiliza para todas las replicaciones el mismo torrente de números seudo-aleatorios ya que el período del generador es lo suficientemente grande (se utiliza la biblioteca estándar *drand48* incluída en el lenguaje C).

```
function MCC(G, sem, N) : {[0..1],real}
      operativos := 0;
      for i:=1 to N do
            sortear (G'i);
            if fi(G'i ) then
                  operativos++;
      end for;
      esp := operativos / N;
      var := esp * (1 - esp) / (N - 1);
      return (esp, var);
end MCC;
```

Algoritmo Monte Carlo crudo

```
procedure sortear (G'i)
      /*sorteo de funcionamiento de nodos*/
      foreach v ∈ V
            sorteo := uniforme[0,1];
            if (sorteo > rv) then
                  deshabilitar v;
      end for;
      /*sorteo de funcionamiento de aristas*/
      foreach e ∈ E
            sorteo := uniforme[0,1];
            if (sorteo > re) then
                  deshabilitar e;
      end for;
end sortear;
```

Sorteo de un estado del grafo

La complejidad del algoritmo es O(|***V***|+|***E***|). Se puede ver observando que se debe realizar un sorteo para cada nodo y arista más una recorrida en profundidad, la cual tiene una complejidad lineal en la cantidad de nodos del grafo.

Algoritmo de Reducción Recursiva de la Varianza

El algoritmo Monte Carlo estándar produce resultados aceptables para redes de mediana confiabilidad, pero para redes altamente confiables se deben realizar muchas replicaciones para obtener estados no operativos (ver observación 2 del Apéndice 1).

En [3] se propone este algoritmo, que reduce el problema original a un problema en una red más pequeña construida a partir de la original, condicionada a uno de sus cortes. El proceso es recursivo y se detiene cuando la red se encuentra siempre en estado operativo o no operativo, independientemente del estado de sus componentes. La construcción del método se realiza para obtener una estimación de la medida $\boldsymbol{Q_k}$ (anti-confiabilidad para un conjunto $\boldsymbol{K}$ de terminales), pero se puede generalizar para todas las restantes.

Definiciones y notación

- Red *K*-conexa: una red $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ con un conjunto de terminales asociado $\boldsymbol{K}\subseteq\boldsymbol{V}$ es *K*-conexa cuando existe por lo menos un camino entre todo par de nodos del conjunto $\boldsymbol{K}$ (red en estado operativo).
- $\boldsymbol{Q}(\boldsymbol{G})$: anti-confiabilidad de $\boldsymbol{G}$ (probabilidad de que la red $\boldsymbol{G}$ no sea *K*-conexa).
- Un conjunto $\boldsymbol{D} \subset \boldsymbol{V}\cup\boldsymbol{E}$ es un *K*-corte extendido de $\boldsymbol{G}$ si la subred $\boldsymbol{G'}=(\boldsymbol{V} - \boldsymbol{D}, \boldsymbol{E} - \boldsymbol{D})$ no es *K*-conexa.
- Para una arista $\boldsymbol{e} \in \boldsymbol{E}$ en $\boldsymbol{G}=(\boldsymbol{V}, \boldsymbol{E})$ con terminales en $\boldsymbol{K}$, $\boldsymbol{G} - \boldsymbol{e}$ es la red cuyo conjunto de nodos es $\boldsymbol{V}$ y cuyo conjunto de aristas se obtiene a partir de $\boldsymbol{E}$ eliminando $\boldsymbol{e}$. El conjunto de terminales de $\boldsymbol{G} - \boldsymbol{e}$ es igual a $\boldsymbol{K}$.
- Para un nodo $\boldsymbol{v} \in \boldsymbol{V}$ en $\boldsymbol{G}=(\boldsymbol{V}, \boldsymbol{E})$ con terminales en $\boldsymbol{K}$, $\boldsymbol{G} - \boldsymbol{v}$ es la red cuyo conjunto de nodos es $\boldsymbol{V} - \{\boldsymbol{v}\}$ y cuyo conjunto de aristas se obtiene a partir de $\boldsymbol{E}$ eliminando todas las aristas incidentes a $\boldsymbol{v}$. El conjunto de terminales de $\boldsymbol{G} - \boldsymbol{v}$ es igual a $\boldsymbol{K} - \{\boldsymbol{v}\}$.
- Sea $\boldsymbol{G}=(\boldsymbol{V}, \boldsymbol{E})$ con terminales en $\boldsymbol{K}$ y $\boldsymbol{d}$ un componente (nodo o arista) de la misma. Se denota $\boldsymbol{G} \mid \boldsymbol{d}$ a la red derivada de $\boldsymbol{G}$ seteando la probabilidad de funcionamiento de $\boldsymbol{d}$ a 1 ($\boldsymbol{d}$ será un componente perfecto).
- Sea $\boldsymbol{G}=(\boldsymbol{V}, \boldsymbol{E})$ con terminales en $\boldsymbol{K}$ y $\boldsymbol{d}$ un componente (nodo o arista) de la misma. Se denota $\boldsymbol{G} * \boldsymbol{d}$ a la red derivada de $\boldsymbol{G}$ seteando la probabilidad de funcionamiento de $\boldsymbol{d}$ a 1. Si luego se halla una arista $\boldsymbol{e}=(\boldsymbol{v_1}, \boldsymbol{v_2})$ (si $\boldsymbol{d}$ es una arista será $\boldsymbol{e}=\boldsymbol{d}$; si $\boldsymbol{d}$ es un nodo será $\boldsymbol{v_1}=\boldsymbol{d}$ ó $\boldsymbol{v_2}=\boldsymbol{d}$) tal que $\boldsymbol{r_e}=\boldsymbol{r_{v1}}=\boldsymbol{r_{v2}}=1$ se debe realizar la contracción de la misma, esto es, eliminando $\boldsymbol{e}$, fundiendo sus extremidades $\boldsymbol{v_1}$ y $\boldsymbol{v_2}$ en un nuevo nodo $\boldsymbol{w}$ y seteando el nuevo conjunto de terminales a $\boldsymbol{K} - \{\boldsymbol{v_1}, \boldsymbol{v_2}\} \cup\{\boldsymbol{w}\}$ si $\boldsymbol{v_1}$ o $\boldsymbol{v_2}$ pertenece a $\boldsymbol{K}$, o simplemente $\boldsymbol{K}$ si ninguno es terminal.

El objetivo ahora, es construir un estimador con igual esperanza que el utilizado en **MCC** y con menor varianza. Para ello se consideran las siguientes propiedades:

Propiedad 1

Sea $\boldsymbol{G}=(\boldsymbol{V}, \boldsymbol{E})$ con terminales en $\boldsymbol{K} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_{|K|}}\}$, entonces se cumple:

$$R(G) = \left(\prod_{v\in K} r_v\right) R(G \mid v_1 \mid v_2 \mid ... \mid v_{|k|})$$

y

$$Q(G) = \left(1 - \prod_{v\in K} r_v\right) + \left(\prod_{v\in K} r_v\right) Q(G \mid v_1 \mid v_2 \mid ... \mid v_{|k|})$$

Significa que la medida de confiabilidad puede obtenerse seteando los nodos terminales como perfectos, multiplicando el resultado de la confiabilidad de esa red resultante por los productos de las confiabilidades de los nodos terminales. Esta propiedad permite reducir el caso de fallas en aristas y nodos al modelo de nodos perfectos.

Propiedad 2

Sea $\boldsymbol{G}$=($\boldsymbol{V}$, $\boldsymbol{E}$) con terminales en $\boldsymbol{K}$, y $\boldsymbol{d}$ un componente de $\boldsymbol{G}$, entonces se cumple $\boldsymbol{R}(\boldsymbol{G} \mid \boldsymbol{d}) = \boldsymbol{R}(\boldsymbol{G} * \boldsymbol{d})$.

Significa que si al setearse un componente como perfecto en la red original se puede realizar contracción, la confiabilidad de la red original es igual a la confiabilidad de la red "contraída". Utilizando esta propiedad, el algoritmo **RVR** (que se presenta más adelante) reduce el problema original a un problema en una red cada vez de menor tamaño.

Las demostraciones de las propiedades 1 y 2 pueden encontrarse en [3].

Si definimos la variable aleatoria $\boldsymbol{Y}(\boldsymbol{G})$ como $\boldsymbol{Y}(\boldsymbol{G})=1 - \phi\ (\boldsymbol{X_G})$ (cuya esperanza es el estimador de la confiabilidad para **MCC**, donde $\boldsymbol{X_G}$ es un vector de los estados aleatorios de los componentes), el objetivo de la técnica RVR es construir una variable aleatoria $\boldsymbol{Z}(\boldsymbol{G})$ con la misma esperanza de $\boldsymbol{Y}(\boldsymbol{G})$ y menor varianza. Dicha variable aleatoria se construye a partir de un *K*-corte extendido $\boldsymbol{D}$, y se expresa en términos de $|\boldsymbol{D}|$ variables aleatorias $\boldsymbol{Y}(\boldsymbol{G_i})$ correspondientes a estados de la red original.

Propiedad 3

Para una red $\boldsymbol{G}$ tal que $\boldsymbol{r_v}=1\ \forall \boldsymbol{v} \in \boldsymbol{K}$ (red donde los terminales no fallan) sean

- $\boldsymbol{D} = \{\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_{|D|}}\}$ un *K*-corte extendido en $\boldsymbol{G}$
- $\boldsymbol{A_D}$ el evento *"todos los componentes en **D** fallan"*
- $\boldsymbol{Q_D} = \Pr\{A_D\} = \prod_{i=1}^{|D|}\left(1 - r_{d_i}\right)$:la probabilidad del evento $\boldsymbol{A_D}$
- $\boldsymbol{B_i}$ el evento *"todos los componentes en $\boldsymbol{D_i} = \{\boldsymbol{d_1}, \boldsymbol{d_2}, ..., \boldsymbol{d_{i-1}}\}$ fallan y $\boldsymbol{d_i}$ funciona"*
- $\boldsymbol{G_i} = (\boldsymbol{G} - \boldsymbol{d_1} - ... - \boldsymbol{d_{i-1}}) * \boldsymbol{d_i}$
- $\boldsymbol{V}$ variable aleatoria discreta independiente de las $\boldsymbol{Y}(\boldsymbol{G_i})$ con

$$\Pr\{V = v\} = \Pr\{B_v\}/\left(1 - Q_D\right) = r_{d_v}\prod_{i=1}^{v-1}\left(1 - r_{d_i}\right)\ /\left(1 - Q_D\right) \text{ para } 1 \le v \le |D|$$

entonces la variable aleatoria

$$Z(G) = Q_D + \left(1 - Q_D\right)\sum_{i=1}^{|D|} 1_{v=i}\, Y\left(G_i\right)$$

verifica

$$\mathrm{E}\,\{\boldsymbol{Z}(\boldsymbol{G})\} = \boldsymbol{Q}(\boldsymbol{G}) \qquad \text{(a)}$$

$$\mathrm{Var}\,\{\boldsymbol{Z}(\boldsymbol{G})\} = (\boldsymbol{Q}(\boldsymbol{G}) - \boldsymbol{Q_D})\boldsymbol{R}(\boldsymbol{G}) \le \boldsymbol{Q}(\boldsymbol{G})\boldsymbol{R}(\boldsymbol{G}) = \mathrm{Var}\,\{\boldsymbol{Y}(\boldsymbol{G})\} \quad \text{(b)}$$

La Propiedad 3 muestra que la variable aleatoria $\boldsymbol{Z}$ tiene la misma esperanza, y menor varianza que la original $\boldsymbol{Y}$. A grandes rasgos, la justificación de la igualdad (a) se obtiene observando que el suceso $\boldsymbol{A_D}$ y los sucesos $\boldsymbol{B_i}$ constituyen una partición del suceso *"la red se encuentra en estado no operativo"* cuya probabilidad $\boldsymbol{Q}$ se desea calcular, y utilizando el teorema de probabilidades totales; la reducción de varianza se puede ver observando que en la estimación de la medida $\boldsymbol{Q}$, siempre se tienen en cuenta estados de falla (suceso $\boldsymbol{A_D}$). Una justificación detallada de estas demostraciones se puede ver en [8] para el caso particular de $\boldsymbol{R_{st}}$).

En base a lo anterior, se construye en forma recursiva la siguiente variable aleatoria $\boldsymbol{F}$:

$$F(G)=\begin{cases}1 & \text{si G no es K - conexo}\\ 0 & \text{si K está formado por un solo nodo (G siempre K - conexo)}\\ Q_D+(1-Q_D)\sum_{i=1}^{|D|}1_{v=i}F(G_i) & \text{de otra forma}\end{cases}$$

y el siguiente algoritmo denominado **RVR** para obtener una muestra de ***F*** para una red ***G*** :

<u>Algoritmo</u> **<u>RVR</u>**

1 - Chequear fin de recursión:
 Si ***G*** es siempre *K*-conexo (|***K***|=1) retornar 0
 Si ***G*** no es *K*-conexo (evaluando ϕ) retornar 1

2 - Encontrar un *K*-corte extendido ***D*** de ***G***: ***D*** = {$\boldsymbol{d_1}$, $\boldsymbol{d_2}$, ..., $\boldsymbol{d_{|D|}}$}

3 - Calcular $\boldsymbol{Q_D}$ (probabilidad de que todos los componentes en ***D*** fallen)

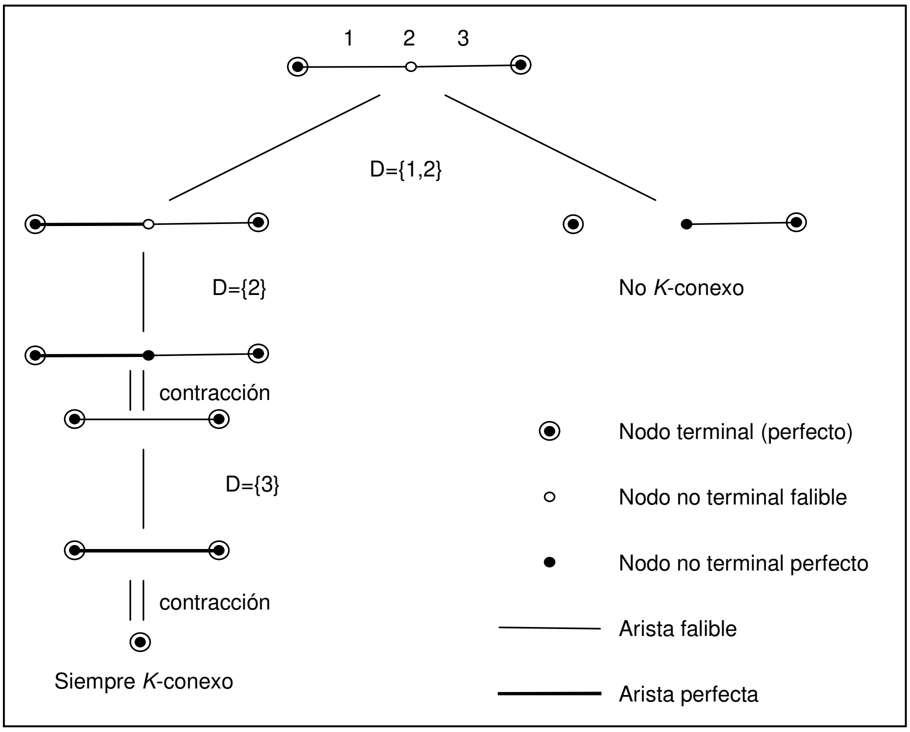
4 - Sortear una muestra ***v*** de la variable aleatoria ***V***

5 - Construir $\boldsymbol{G_V}$ = ($\boldsymbol{G}$ - $\boldsymbol{d_1}$ - ... - $\boldsymbol{d_{v-1}}$) * $\boldsymbol{d_v}$ (eliminaciones y contracción)

6 - Paso recursivo: retornar $\boldsymbol{Q_D}$ + (1 - $\boldsymbol{Q_D}$)***RVR***($\boldsymbol{G_V}$)

Ejemplo

El siguiente ejemplo muestra la ejecución del algoritmo **RVR** en una red con tres nodos (de los cuales dos son terminales) y dos aristas. La computación se muestra en forma de árbol, donde los hijos de un nodo interno son los posibles resultados del sorteo de la variable aleatoria ***V*** (los sucesos $\boldsymbol{B_i}$). Las hojas del árbol corresponden a situaciones donde se cumple alguna condición de parada (red siempre *K*-conexa o no *K*-conexa). Una replicación del algoritmo solamente genera un camino desde la raíz hasta alguna hoja.



Posibles caminos de ejecución de una replicación de RVR para una red particular

Implementación

El algoritmo **RVR** se implementó en una clase denominada **RVR** que hereda las características de **GrafoEstocastico** y de **Medidas** junto con todas las operaciones auxiliares necesarias. A continuación se describen los detalles de implementación del mismo:

```
function RVR(G) : [0..1]
      if (terminales = 1) then
            return 0;
      else
            if not fi(G) then
                  return 1;
            else
                  D := darCorteExtendido(G);
                  Qd := probFallaTodos(D);
                  indice := sortearFunciona(D);
                  c := D[indice];
                  quitar(G, D, indice - 1);
                  agregar(G, c);
                  return Qd + (1 - Qd)* RVR(G);
end RVR;
```

Implementación del algoritmo RVR

La variable `terminales` lleva la cuenta de la cantidad de terminales en la red (la misma se decrementa cuando se eliminan nodos terminales y eventualmente cuando ocurren contracciones) y la función ***fi*** es la heredada de la clase **Medidas**.

Para la estructura del corte extendido se utilizó un arreglo con tope donde en cada posición se almacena un objeto de tipo componente (que puede ser un nodo o una arista) junto con su probabilidad correspondiente a la distribución de la variable aleatoria ***V*** (a utilizar por la función `sortearFunciona`). A continuación se describen las características principales de las funciones auxiliares:

`darCorteExtendido`: Selecciona arbitrariamente un nodo terminal de la red y considera todos los nodos adyacentes y aristas incidentes al mismo con probabilidad de funcionamiento estrictamente menor que 1. Estos componentes son agregados al resultado ubicando primero las aristas y luego los nodos (en particular, en los pasos 4 y 5 del algoritmo **RVR** interesa poder eliminar primero todas las aristas, pues si se eliminan primero los nodos pueden quedar aristas funcionando con alguno de sus extremos fallando, hecho que no afecta el correcto funcionamiento del algoritmo pero puede ser una posible causa de pérdida de eficiencia en la implementación). Finalmente se calculan las probabilidades de los componentes en el corte para el sorteo de la variable aleatoria ***v***.

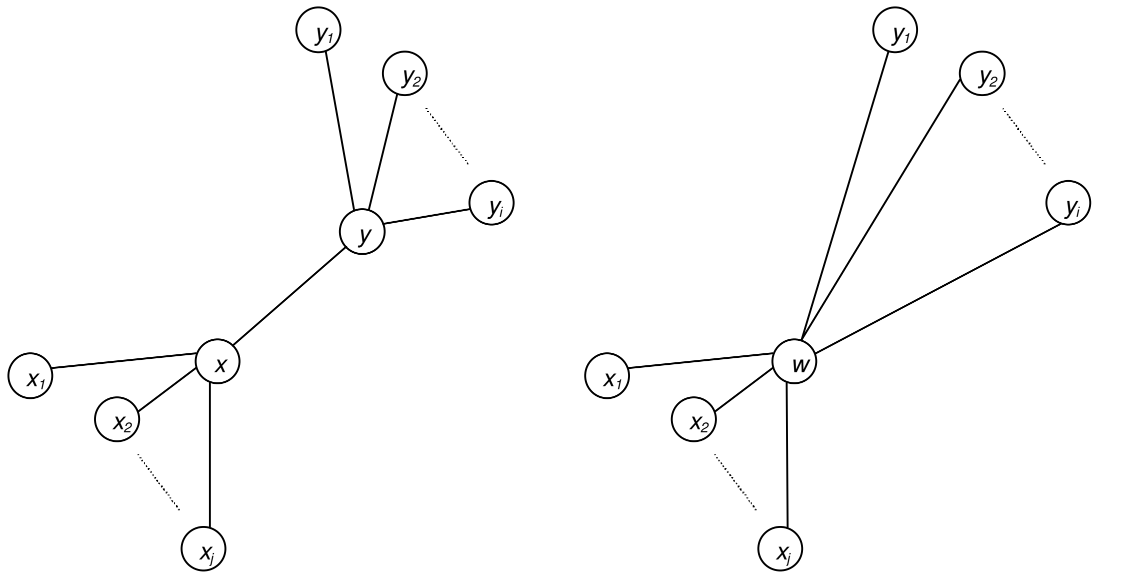
`probFallaTodos`: Simplemente calcula el producto de los complementos de las probabilidades de funcionamiento de cada uno de los componentes del corte extendido.

`sortearFunciona`: Obtiene una muestra de la variable aleatoria ***v*** en base al sorteo de una v.a. uniforme en el intervalo [0,1]. Devuelve el índice del componente sorteado en el corte.

`quitar`: Elimina de la red ***G*** todos los componentes del corte hasta la posición `índice` - 1. La eliminación de un componente se efectúa marcándolo como deshabilitado y en el caso de ser un nodo terminal se decrementa el valor de la variable `terminales`.

`agregar`: Setea la probabilidad de funcionamiento del componente a 1 y chequea si es necesario realizar una contracción.

La operación de contracción se realiza cuando se encuentra una arista con probabilidad de funcionamiento 1 cuyos nodos extremos también tienen probabilidad de funcionamiento 1. El proceso de contracción se describe en la siguiente figura:



Grafo antes y después de la contracción de la arista (x , y)

En el caso en que uno de los nodos sea terminal se considera a ***w*** como tal y se decrementa el valor de la variable `terminales`. Si los dos son terminales se elige uno arbitrariamente como nodo ***w***. Un caso especial se da cuando ya existe una arista entre el nodo ***x*** y un adyacente a ***y*** (sea $\boldsymbol{y_i}$ tal nodo, $\boldsymbol{r_1}$ su probabilidad de funcionamiento y $\boldsymbol{r_2}$ la probabilidad de funcionamiento de la arista (***y***, $\boldsymbol{y_i}$)). En este caso, al realizar la contracción se debe efectuar una reducción en paralelo ([7]) de las dos aristas (sustituyéndose las dos originales por una sola) donde la probabilidad de funcionamiento de la nueva arista será $\boldsymbol{r_1 + r_2 - r_1 * r_2}$.

Se puede ver que en cada paso recursivo, el problema original se reduce a uno (y solo uno) más pequeño, ya que en el paso 5 del algoritmo se reduce la cantidad de componentes de la red original (mediante eliminaciones y contracciones). Así, la cantidad de invocaciones recursivas se puede acotar por |***V***|+|***E***| y observando que la operación más costosa en cada paso es la evaluación de la función de estructura ***fi*** (de orden O(|***V***|)), el orden del algoritmo **RVR** tendrá orden de complejidad O(|***V***| * (|***V***|+|***E***|)).

Para realizar varias replicaciones de la simulación, el algoritmo anterior es invocado varias veces mediante una iteración. La esperanza de ***F***(***G***) se estima como:

$$E(F) = \frac{1}{N} \sum_{i=1}^{N} F_i$$

y la varianza se estima mediante la siguiente expresión

$$Var(F) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \left(F_i - E(F) \right)^2$$

que por comodidad para su cálculo se puede llevar a

$$Var(F) = \frac{1}{N(N-1)}\left(\sum_{i=1}^{N} F_i^2 - \frac{1}{N}\left(\sum_{i=1}^{N} F_i\right)^2\right)$$

El algoritmo implementado para el cálculo de la confiabilidad utilizando **RVR** recibe el grafo, un identificador de semilla para el generador de números seudo-aleatorios (se utiliza de la misma forma que en **MCC**) y una cantidad ***N*** de replicaciones.

```
function conf(G, sem, N):{[0..1],real}
      s := 0;  /* acumula los resultados de cada replicacion */
      ss := 0; /* acumula los cuadrados de cada replicacion */
      setearSemilla(sem);
      for i:=1 to N do
            G' := G;
            x := 1 - RVR(G');
            /* RVR calcula la anti-confiabilidad */
            s := s + x;
            ss := ss + x * x;
      end for;
      esp := s / N;
      var := (1/(N*(N-1)))*(ss - s*s/N);
      return (esp, var);
end conf;
```

Cálculo de la confiabilidad de una red utilizando RVR

En los siguientes dos capítulos se presentan los casos de prueba para la validación y comparación de la eficiencia de los tres algoritmos implementados.

4 - Pruebas de validación

En este capítulo se presentan los casos de prueba que se utilizaron para la validación de los algoritmos estudiados. Para cada caso se presentan las topologías, los valores seleccionados para las confiabilidades de los componentes y los resultados obtenidos.

El objetivo de estas pruebas es obtener información útil para la validación de la implementación de los algoritmos estudiados. Se asume para esta parte que el software ha cumplido la etapa de verificación, por lo tanto está libre de errores de tiempo de ejecución, lo que no significa que esté libre de errores lógicos (diferencias entre la estrategia propuesta y la implementada).

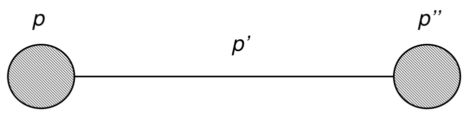
La estrategia propuesta para las mismas es la siguiente:

- Considerar redes de tamaño relativamente pequeño (confiables y no confiables).
- Calcular el índice de confiabilidad de forma manual (utilizando por ejemplo el algoritmo de enumeración completa de estados, o en casos en que sea posible, en forma analítica).
- Comparar con los resultados proporcionados por los algoritmos.

Redes consideradas

Caso 1

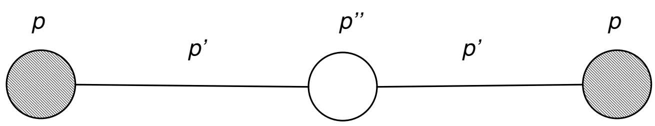
Grafo compuesto por dos nodos terminales y una única arista que los une. En este caso interesa la medida $\boldsymbol{R_{st}}$ que se calcula como *pp'p''*.



Grafo completo de 2 nodos

Caso 2

Grafo compuesto por tres nodos de los cuales dos son terminales y dos aristas. La medida $\boldsymbol{R_{st}}$ para el mismo se calcula como $p^2p'^2p''$.

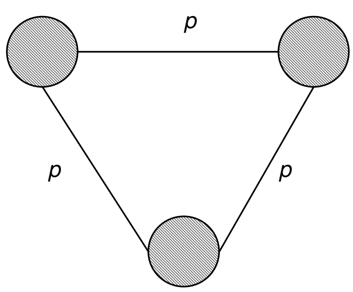


Grafo de 3 nodos

Para todas las pruebas siguientes se considerarán terminales perfectos (probabilidad de funcionamiento igual a 1).

Caso 3: K_3

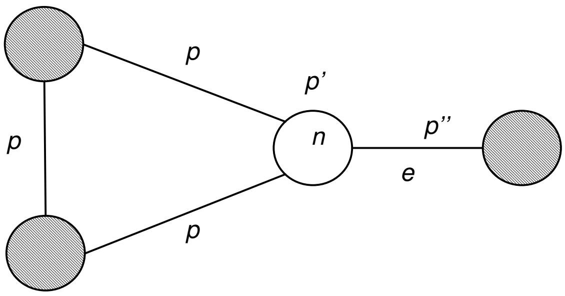
Grafo completo de 3 nodos, donde todos son terminales (en este caso interesa la medida $\boldsymbol{R_v}$). Se considera una misma probabilidad *p* de funcionamiento para todas las aristas. Un grafo completo con probabilidades altas de funcionamiento tanto en aristas como en nodos es una red altamente confiable. La confiabilidad para este grafo se puede calcular en forma analítica mediante la expresión $p^3 + 3(1 - p)p^2$, donde el primer término de la suma corresponde a la probabilidad del suceso *"todas las aristas funcionan"* y el segundo a la probabilidad del suceso *"todas las aristas funcionan menos una"* (para cada arista).



Grafo completo de 3 nodos

Caso 4: Red con puente y punto de articulación

Se considera la siguiente red (donde interesa la medida $\boldsymbol{R_k}$) que presenta un puente (arista *e*) y un punto de articulación (nodo *n*).

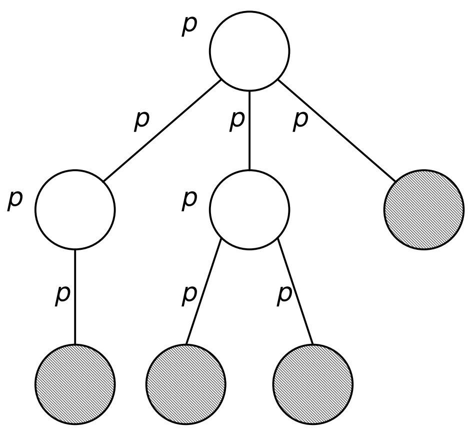


Red con puente y punto de articulación

La confiabilidad de esta red depende fuertemente de las probabilidades de funcionamiento del nodo *n* y de la arista *e*, como se ve reflejado en la siguiente expresión obtenida para su cálculo en forma analítica: $\boldsymbol{R_k} = p''p'[p^3 + 3(1 - p)p^2]$

Caso 5: Arbol

Se considera un árbol donde los nodos terminales son las hojas (en este caso interesa la medida $\boldsymbol{R_k}$), con probabilidad de funcionamiento *p* para todas las aristas y nodos que no son terminales. Un árbol tendrá valores bajos de confiabilidad pues la falla de cualquier nodo o arista desconecta cualquier par de nodos de la red. La confiabilidad de una red con topología de árbol se calcula como el producto de las probabilidades de funcionamiento de cada uno de sus componentes.

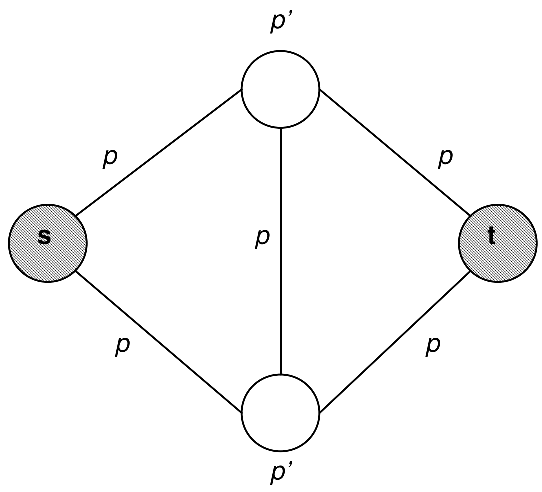


Arbol

Caso 6: Grafo puente

En este caso la medida de interés es $\boldsymbol{R_{st}}$. La confiabilidad de esta red depende fuertemente del valor de *p*', que es la probabilidad de funcionamiento de los nodos intermedios en los caminos de ***s*** a ***t***. La expresión analítica obtenida para su cálculo es la siguiente:

$\boldsymbol{R_{st}} = pp'p + (1-p)pp'p + (1-p')ppp'p + (1-p)p'ppp'p + pp'(1-p)(1-p)p'pp + (1-p)p'ppp'(1-p)p$



Grafo puente

Resultados

Los casos de prueba fueron generados en la herramienta HEIDI [9] y se implementó una rutina en la clase **GrafoEstocastico** que carga un grafo desde un archivo con el formato de exportación de HEIDI (archivo de texto). Esto facilitó considerablemente el proceso de generación de los casos de prueba.

Los algoritmos implementados trabajan con números reales en doble precisión (tipo `double` del lenguaje C) los cuales tienen hasta 15 dígitos significativos. Los cálculos exactos basados en expresiones analíticas se realizaron en forma manual utilizando la herramienta Microsoft Excel (en la que se puede trabajar con la misma precisión que en C).

Se realizaron pruebas con valores particulares para las probabilidades de funcionamiento de nodos y aristas. Los algoritmos **MCC** y **GCE** no requirieron de un trabajo fuerte de depuración mientras que para el algoritmo **RVR**, se detectaron errores en las pruebas de validación que debieron ser corregidos.

Para los algoritmos estimativos se realizaron 10^4 replicaciones. Los resultados finales obtenidos se muestran en la siguiente tabla (página siguiente).

		Analítico	GCE	MCC		RVR	
				R	**Var**	***R***	**Var**
Caso 1 *p*=*p*'=*p*''=0,9		0,729	0,729	0,728	2,0E-05	0,729	3,1E-17
Caso 2 *p*=*p*'=*p*''=0,9		0,59049	0,59049	0,586	2,4E-05	0,589	3,6E-06
Caso 3 *p*=0,9		0,972	0,972	0,970	2,9E-06	0,9718	6,7E-08
Caso 4 *p*=0,9	*p*'=0,5 *p*''=0,9	0,4374	0,4374	0,437	2,5E-05	0,4369	3,7E-07
	p'=0,9 *p*''=0,5	0,4374	0,4374	0,430	2,5E-05	0,4365	4,6E-07
Caso 5 *p*=0,9		0,387420489	0,387420489	0,385	2,4E-05	0,380	1,5E-05
Caso 6 *p*=0,9	*p*'=0,5	0,64962	0,64962	0,644	2,3E-05	0,646	6,8E-06
	p'=0,9	0,9383688	0,9383688	0,933	6,2E-06	0,936	2,0E-06

Resultados de las pruebas de validación

Interpretación de resultados

Tomando como referencia los valores calculados en forma analítica (primer columna de la tabla), se puede observar que los resultados del algoritmo **GCE** coinciden con los anteriores para todos los casos.

Para los algoritmos estimativos, se observa en general la mayor precisión (y menor varianza) de **RVR** por sobre **MCC**.

5 - Pruebas de eficiencia

En este capítulo se presentan los casos de prueba que se utilizaron para realizar la comparación de la eficiencia de los algoritmos estudiados. Para cada caso se presentan las topologías, los conjuntos de valores seleccionados para las confiabilidades de los componentes, los resultados y conclusiones obtenidos, y algunas comparaciones con resultados obtenidos en trabajos anteriores.

Los resultados a comparar son básicamente los tiempos de ejecución, y para los algoritmos estimativos la varianza de la muestra. La estrategia propuesta para las pruebas de comparación de eficiencia es la siguiente:

- Considerar redes de tamaño mediano y relativamente grande (para las cuales los algoritmos exactos son impracticables por su excesivo tiempo de ejecución).
- Incluir en la medida de lo posible, redes utilizadas en otros estudios similares.
- Correr el algoritmo **GCE** cuando es factible.
- Correr los algoritmos **MCC** y **RVR** con dichas redes, tomar los valores de confiabilidad, varianza y tiempos de ejecución.
- Utilizar una medida basada en los valores anteriores para la comparación de la eficiencia de los algoritmos.

Plan de pruebas

Se han seleccionado tanto un conjunto de topologías como también una serie de valores para las probabilidades de funcionamiento de sus componentes. Dicha elección tiene en cuenta dos objetivos principales:

- Estudiar la evolución de la confiabilidad y su error en función de los valores de las probabilidades de funcionamiento de sus componentes.
- Estudiar la evolución de los tiempos de ejecución de los algoritmos en base a los tamaños de las redes.

Las topologías seleccionadas son las siguientes:

1 - ARPANET
2 - Grafo bipartito completo con fuente y terminal
3 - Red telefónica de fibra óptica de Montevideo
4 - Dodecaedro
5 - Malla dispersa
6 - Malla densa

A continuación se describen en detalle cada una de las topologías, las pruebas realizadas y los resultados obtenidos. Estos se presentan en tablas conteniendo valores de confiabilidad, varianza, tiempos de ejecución y eficiencia relativa (producto de los cocientes de varianzas y tiempos: W = Var * T). Esta última medida se utiliza para comparar la performance general entre algoritmos.

Para todos los casos de prueba, los algoritmos se corrieron en una estación de trabajo Sun sparc Enterprise 250 con 256 Mb de memoria RAM y procesador de 250 Mhz, bajo el sistema operativo SunOS 5.7, con relativamente baja carga de trabajos. Los fuentes fueron compilados utilizando el compilador C++ de GNU (*gcc* versión 2.95) con la opción de optimización de código de 64 bits. Los casos de prueba se generaron de la misma forma que para las pruebas de validación. Los algoritmos estimativos se corrieron con 10^4 replicaciones (salvo indicación). Los resultados de tiempos de ejecución se muestran en segundos.

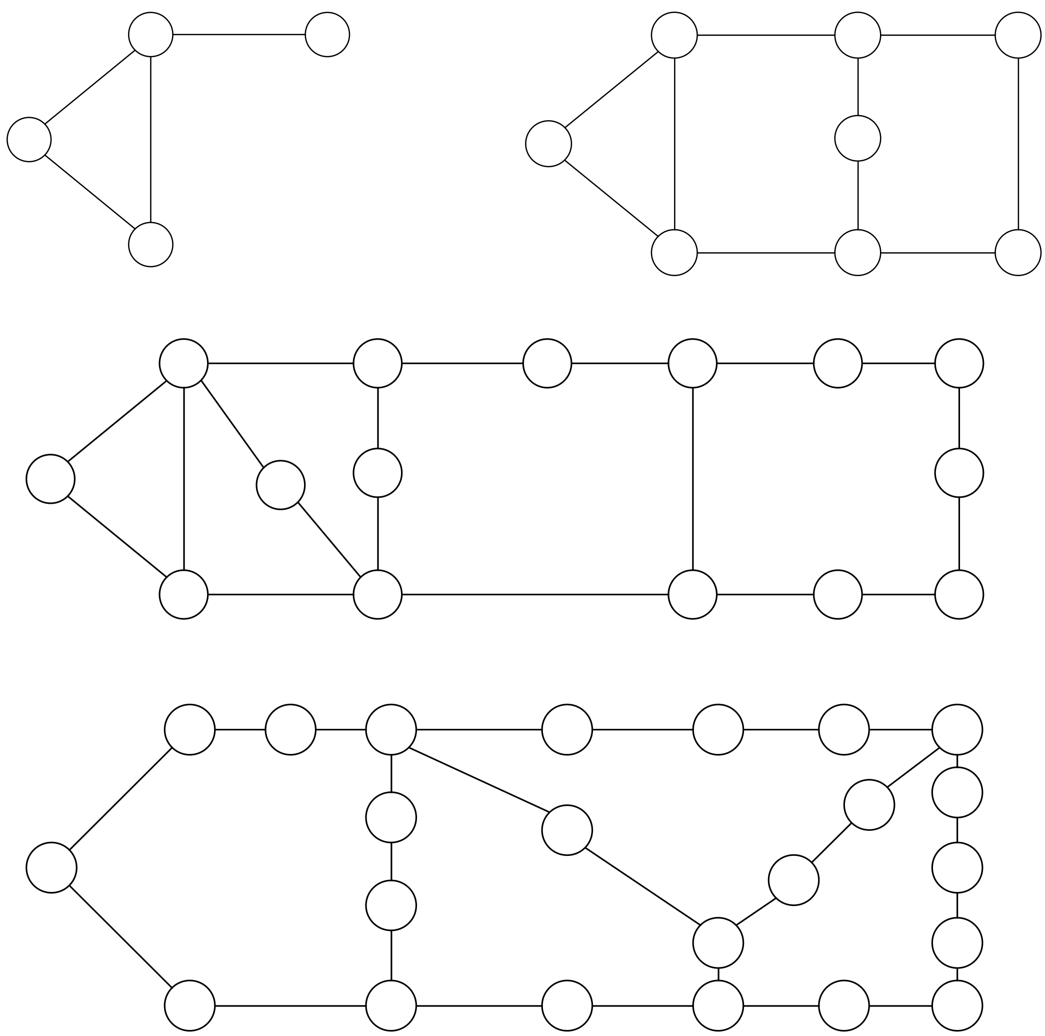
Casos de prueba

Caso 1: ARPANET

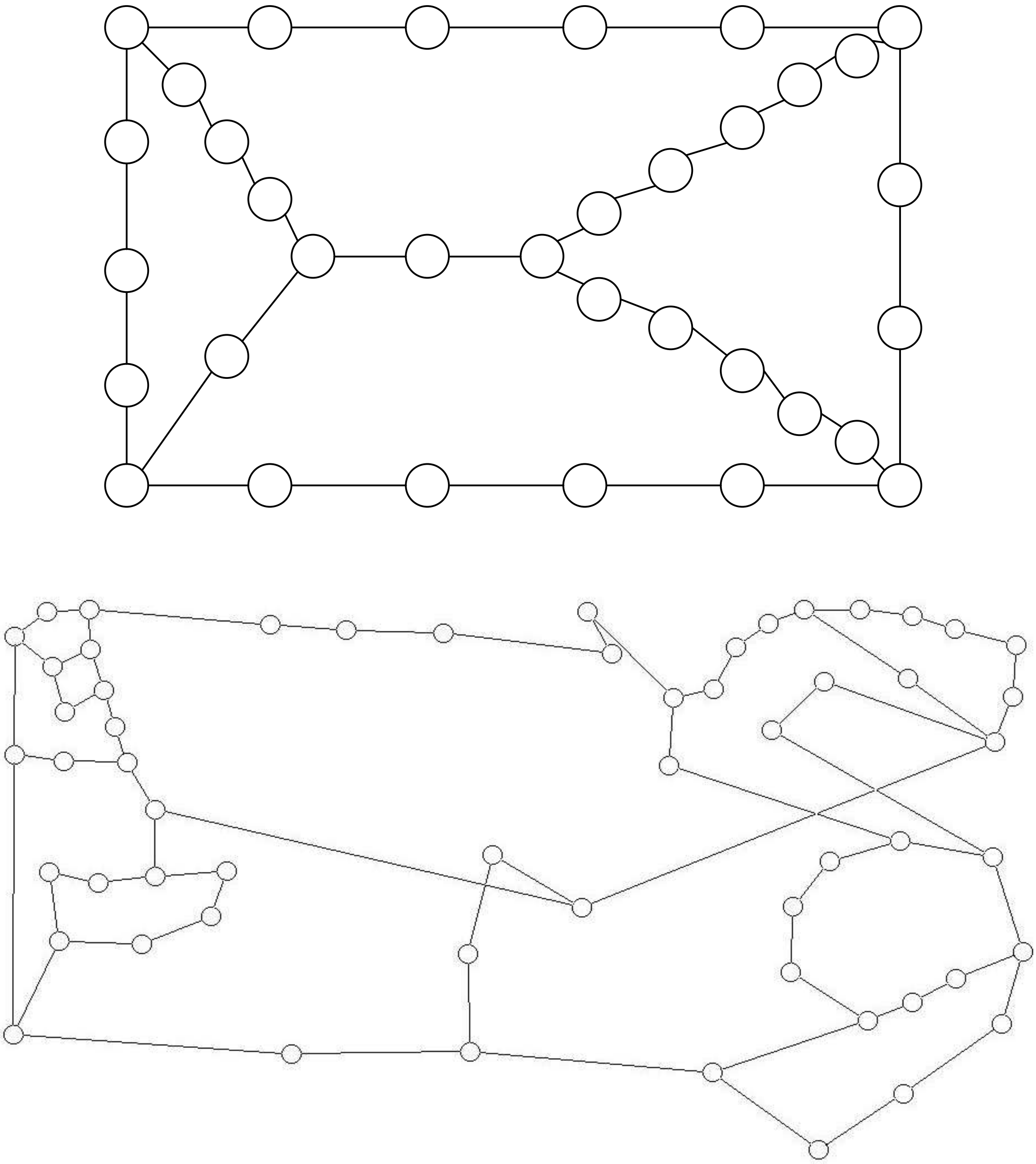
Topología de la red de computadoras que dio origen a la actual Internet. El objetivo principal de este caso de prueba es comparar la evolución de los tiempos de ejecución de cada algoritmo en relación al tamaño de la red así como determinar el tamaño aproximado de una red para la cual es factible utilizar el algoritmo **GCE**. Las topologías fueron tomadas de [12] y [2] y constituyen la evolución de la red desde su surgimiento en 1969 hasta 1979. Se realizaron pruebas con valores 0,9 para las probabilidades de funcionamiento de nodos y aristas. La medida de interés en este caso es $\boldsymbol{R_V}$. Las características de cada topología (cantidad de nodos y aristas) se muestran en la siguiente tabla.

Topología	|V|	|E|
Dic. 69	4	4
Jul. 70	8	10
Mar. 71	15	19
Abr. 72	23	26
Set. 72	34	38
Año 79	58	69

Topologías de ARPANET



Evolución de ARPANET (Diciembre de 1969, Julio de 1970, Marzo de 1971 y Abril de 1972)

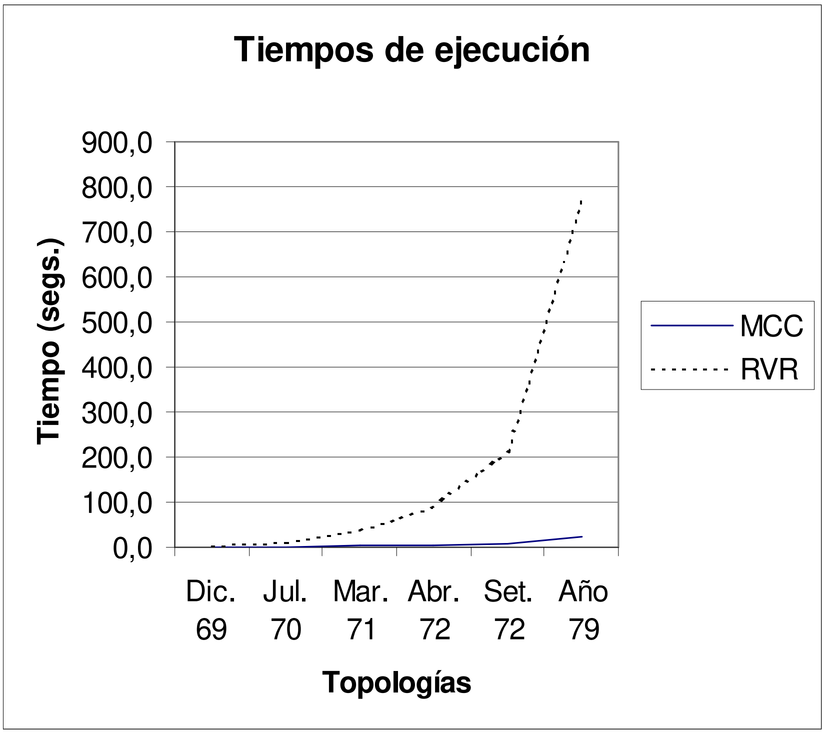


Evolución de ARPANET (Setiembre de 1972 y Año 1979)

Resultados

Para la tercer topología de este juego (ARPANET de Marzo de 1971, con 15 nodos y 19 aristas) fue imposible correr el algoritmo **GCE** (se ejecutó durante más de 12 horas sin obtener el resultado).

	GCE		MCC			RVR			Var_{MCC}/ Var_{RVR}	T_{MCC} / T_{RVR}	W_{MCC} / W_{RVR}
	R	T_{GCE}	*R*	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
Dic. 69	0,573956	0,02	0,57048	2,5E-05	0,6	0,57332	3,1E-07	3,5	7,8E+01	1,8E-01	1,4E+01
Jul. 70	0,401076	5,69	0,40111	2,4E-05	1,4	0,40138	1,8E-07	11,8	1,4E+02	1,2E-01	1,6E+01
Mar. 71	-	-	0,16945	1,4E-05	3,1	0,16767	1,6E-07	37,3	9,0E+01	8,2E-02	7,4E+00
Abr. 72	-	-	0,05934	5,6E-06	5,3	0,05866	7,7E-08	89,6	7,2E+01	5,9E-02	4,2E+00
Set. 72	-	-	0,01458	1,4E-06	9,6	0,01464	1,4E-08	211,6	1,0E+02	4,5E-02	4,6E+00
Año 79	-	-	0,00095	9,5E-08	24,2	0,00095	1,0E-10	772,4	9,1E+02	3,1E-02	2,8E+01



Tiempos de ejecución para las topologías de ARPANET

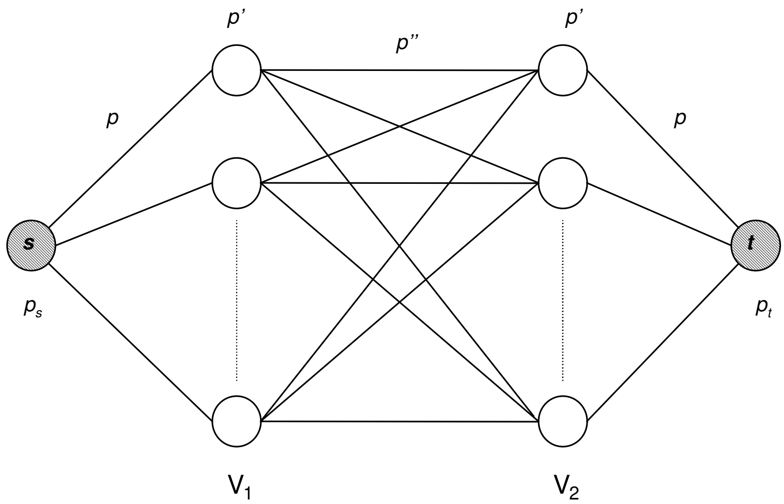
Conclusiones

- Se observa como es de esperarse un mayor crecimiento de los tiempos de ejecución de **RVR** con respecto a **MCC**, sin embargo las varianzas (ver tabla) son siempre menores para el primero.
- Las eficiencias relativas muestran siempre un mayor trabajo para **MCC** (debido a los valores altos de las varianzas).
- El algoritmo **GCE** resulta impracticable para una red con 15 nodos y 19 aristas (y por lo tanto, también para redes de mayor tamaño).

Para las siguientes pruebas se tiene en cuenta esta última observación ya que todas las topologías tienen mayor número de componentes.

Caso 2: Grafo bipartito completo con fuente y terminal

Se considera un tipo particular de grafo de conectividad relativamente alta, donde interesa la medida R_{st}. La topología consiste en un grafo bipartito completo más dos terminales, cada uno de los cuales es adyacente a todos los nodos de una partición. Los objetivos de este caso de prueba son los mismos que para el caso anterior. También interesa estudiar la evolución de la confiabilidad con el aumento de las cardinalidades de las clases V_1 y V_2 de la bipartición.



Grafo bipartito completo

Se consideran cuatro pruebas distintas variando las cardinalidades de las clases de la bipartición, todas con probabilidades $p=p'=p''=p_s=p_t=0{,}9$. En la siguiente tabla se muestran las características de cada caso de prueba.

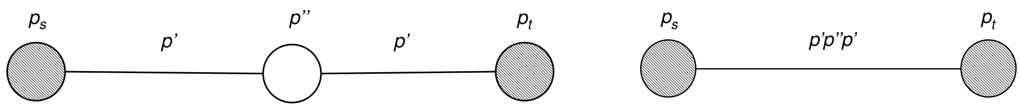
	$\|V_1\|$	$\|V_2\|$	$\|E\|$
Bip. 1	2	10	32
Bip. 2	5	10	65
Bip. 3	10	10	20
Bip. 4	20	10	230

Resultados

	MCC			RVR			Var_{MCC}/Var_{RVR}	T_{MCC}/T_{RVR}	W_{MCC}/W_{RVR}
	R	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
Bip. 1	0,781	1,7E-05	3,4	0,77977877	1,2E-06	71,7	1,4E+01	4,8E-02	6,6E-01
Bip. 2	0,810	1,5E-05	5,3	0,80992153	6,1E-10	120,9	2,5E+04	4,4E-02	1,1E+03
Bip. 3	0,81	1,5E-05	8,4	0,80999995	2,9E-16	231,4	5,3E+10	3,6E-02	1,9E+09
Bip. 4	0,81	1,5E-05	15,9	0,80999993	3,6E-16	598,3	4,3E+10	2,7E-02	1,2E+09

Se puede observar que en los últimos dos casos los valores de confiabilidad obtenidos mediante **MCC** son iguales para topologías distintas (lo que no ocurre con **RVR**). Esto se puede justificar observando que el factor dominante en la confiabilidad de la red es la confiabilidad de los terminales (0,9 en este caso) ya que la topología de la red es muy densa (tanto en la comunicación entre nodos intermedios como en la comunicación entre los terminales y nodos intermedios) lo que hace que tenga un valor de confiabilidad muy cercano a 1 (si los terminales fueran perfectos).

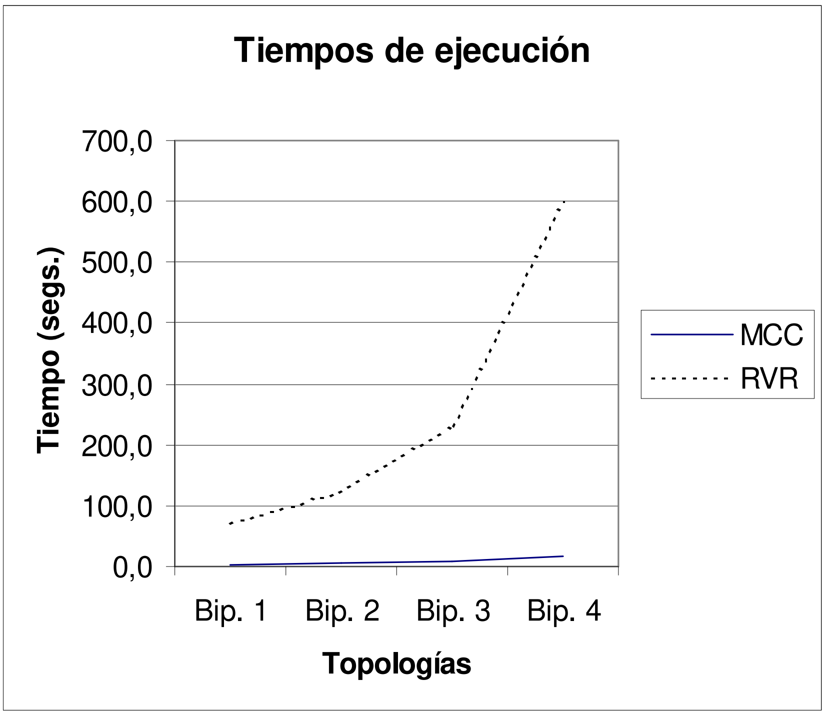
Una forma de ver lo anterior es condensando la subred formada por los nodos intermedios en un solo nodo con probabilidad de funcionamiento alta (*p''* en figura siguiente). Las aristas que unen los terminales con los nodos intermedios también se pueden condensar en aristas con probabilidades también altas (*p'*). La confiabilidad de la primer red de la figura es aproximadamente la misma que la de la segunda, donde R_{st} se calcula como $p_s p'p''p'p_t$ y si el producto $p'p''p'$ es cercano a 1, la confiabilidad estará determinada por el producto $p_s p_t$.



Grafo bipartito completo con fuente y terminal "condensado"

Una prueba para las topologías Bip. 3 y Bip. 4 con probabilidades 0,9 y 0,5 para *s* y *t* respectivamente, valida la observación formulada anteriormente. Los resultados obtenidos son los siguientes.

	MCC			RVR		
	R	**Var_{MCC}**	**T_{MCC}**	***R***	**Var_{RVR}**	**T_{RVR}**
Bip. 3	0,45	2,5E-05	8,3	0,449999972	9,0E-17	229,5
Bip. 4	0,45	2,5E-05	16,0	0,449999963	1,1E-16	600,7



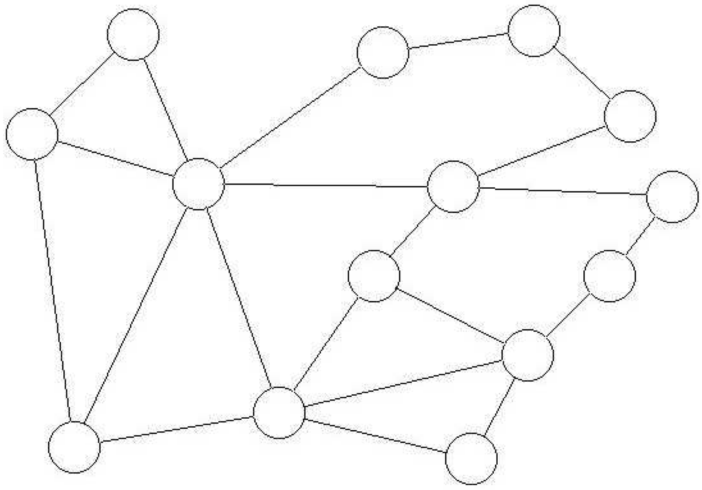
Tiempos de ejecución para las topologías de grafos bipartitos completos

Conclusiones

- En general se pueden hacer las mismas observaciones que para el caso de prueba anterior.
- Se puede ver la mayor precisión de **RVR** en este tipo de topologías en los resultados para los dos últimos casos, donde **MCC** obtiene el mismo resultado para Bip. 3 y Bip 4.

Caso 3: Red telefónica de fibra óptica de Montevideo (1992)

Esta topología es una versión parcial de la red telefónica de fibra óptica de Montevideo (año 1992). Los datos son tomados de [10], donde se cuenta con resultados numéricos. Está formada por 14 nodos y 21 aristas. La medida de interés en este caso es R_V.



El principal objetivo de este caso de prueba es realizar comparaciones con resultados obtenidos en trabajos anteriores. Se realizaron en total 36 pruebas variando las probabilidades de funcionamiento de nodos y aristas.

En lo que sigue:

- p_n: confiabilidad de nodos
- p_e: confiabilidad de aristas

Resultados

Prueba		MCC			RVR			Var_{MCC} / Var_{RVR}	T_{MCC} / T_{RVR}	W_{MCC} / W_{RVR}
		R	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
p_n=0,9	p_e=0,9	0,2051	1,6E-05	3,0	0,20536713	2,9E-07	34,5	5,6E+01	8,7E-02	4,9E+00
p_n=0,95	p_e=0,9	0,4372	2,5E-05	3,0	0,43779045	1,3E-06	34,5	1,9E+01	8,8E-02	1,6E+00
p_n=0,98	p_e=0,9	0,6757	2,2E-05	3,0	0,67655151	3,2E-06	34,5	6,9E+00	8,7E-02	6,0E-01
p_n=0,99	p_e=0,9	0,7789	1,7E-05	3,0	0,77988135	4,2E-06	34,5	4,1E+00	8,7E-02	3,6E-01
p_n=0,999	p_e=0,9	0,8841	1,0E-05	3,0	0,88522291	5,4E-06	34,5	1,9E+00	8,7E-02	1,6E-01
p_n=1,0	p_e=0,9	0,8966	9,3E-06	3,0	0,89770948	5,6E-06	34,5	1,7E+00	8,7E-02	1,5E-01
p_n=0,9	p_e=0,95	0,2222	1,7E-05	3,1	0,22277369	8,3E-08	34,8	2,1E+02	8,8E-02	1,8E+01
p_n=0,95	p_e=0,95	0,4737	2,5E-05	3,1	0,47489679	3,8E-07	34,7	6,6E+01	8,8E-02	5,8E+00
p_n=0,99	p_e=0,95	0,7321	2,0E-05	3,1	0,73389481	9,0E-07	34,8	2,2E+01	8,8E-02	1,9E+00
p_n=0,99	p_e=0,95	0,8439	1,3E-05	3,1	0,84598270	1,2E-06	34,8	1,1E+01	8,8E-02	9,6E-01
p_n=0,999	p_e=0,95	0,9579	4,0E-06	3,1	0,96025283	1,5E-06	34,8	2,6E+00	8,8E-02	2,3E-01
p_n=1,0	p_e=0,95	0,9714	2,8E-06	3,1	0,97379773	1,6E-06	34,8	1,7E+00	8,8E-02	1,5E-01
p_n=0,9	p_e=0,98	0,2276	1,8E-05	3,1	0,22776145	1,5E-08	34,8	1,2E+03	8,8E-02	1,1E+02
p_n=0,95	p_e=0,98	0,4853	2,5E-05	3,1	0,48552943	6,6E-08	34,8	3,8E+02	8,8E-02	3,3E+01
p_n=0,98	p_e=0,98	0,7499	1,9E-05	3,1	0,75032625	1,6E-07	34,8	1,2E+02	8,9E-02	1,1E+01
p_n=0,99	p_e=0,98	0,8645	1,2E-05	3,1	0,86492372	2,1E-07	34,8	5,6E+01	8,8E-02	4,9E+00
p_n=0,999	p_e=0,98	0,9813	1,8E-06	3,1	0,98175227	2,7E-07	34,9	6,8E+00	8,8E-02	6,0E-01
p_n=1,0	p_e=0,98	0,9951	4,9E-07	3,1	0,99560044	2,8E-07	34,8	1,7E+00	8,8E-02	1,5E-01
p_n=0,9	p_e=0,99	0,2285	1,8E-05	3,1	0,22849231	4,2E-09	34,9	4,2E+03	8,9E-02	3,7E+02
p_n=0,95	p_e=0,99	0,4871	2,5E-05	3,1	0,48708743	1,9E-08	34,8	1,3E+03	8,8E-02	1,2E+02
p_n=0,98	p_e=0,99	0,7527	1,9E-05	3,1	0,75273396	4,5E-08	34,8	4,1E+02	8,9E-02	3,6E+01
p_n=0,99	p_e=0,99	0,8677	1,1E-05	3,1	0,86769915	6,0E-08	34,9	1,9E+02	8,9E-02	1,7E+01
p_n=0,999	p_e=0,99	0,9849	1,5E-06	3,1	0,98490260	7,8E-08	34,9	1,9E+01	8,8E-02	1,7E+00
p_n=1,0	p_e=0,99	0,9988	1,2E-07	3,1	0,99879520	8,0E-08	34,9	1,5E+00	8,8E-02	1,3E-01
p_n=0,9	p_e=0,999	0,2288	1,8E-05	3,1	0,22876699	5,7E-15	34,8	3,1E+09	8,9E-02	2,7E+08
p_n=0,95	p_e=0,999	0,4877	2,5E-05	3,1	0,48767298	2,6E-14	34,9	9,6E+08	8,9E-02	8,5E+07
p_n=0,98	p_e=0,999	0,7536	1,9E-05	3,1	0,75363885	6,2E-14	34,9	3,0E+08	8,8E-02	2,6E+07
p_n=0,99	p_e=0,999	0,8687	1,1E-05	3,1	0,86874225	8,3E-14	34,9	1,4E+08	8,9E-02	1,2E+07
p_n=0,999	p_e=0,999	0,9861	1,4E-06	3,1	0,98608659	1,1E-13	34,9	1,3E+07	8,9E-02	1,1E+06
p_n=1,0	p_e=0,999	1	*	3,1	0,99999590	1,1E-13	34,9	**	8,8E-02	**
p_n=0,9	p_e=1,0	0,2288	1,8E-05	3,1	0,22876792	*	7,1	**	4,3E-01	**
p_n=0,95	p_e=1,0	0,4877	2,5E-05	3,1	0,48767498	*	7,1	**	4,3E-01	**
p_n=0,98	p_e=1,0	0,7536	1,9E-05	3,1	0,75364194	*	7,1	**	4,3E-01	**
p_n=0,99	p_e=1,0	0,8687	1,1E-05	3,1	0,86874581	*	7,1	**	4,3E-01	**
p_n=0,999	p_e=1,0	0,9861	1,4E-06	3,1	0,98609064	*	7,1	**	4,3E-01	**
p_n=1,0	p_e=1,0	1	*	3,1	1	*	7,1	**	4,3E-01	**

Los casilleros marcados con * corresponden a valores nulos obtenidos para la varianza.
En los casilleros marcados con ** no tiene sentido calcular los cocientes al contarse con numeradores nulos.

Conclusiones

- Las mismas observaciones realizadas para los casos anteriores en cuanto a varianzas y tiempos de ejecución son aplicables a este caso.
- Se puede observar en este caso que para redes muy confiables (ver fila 30, red con nodos perfectos y aristas con confiabilidades de 0,999), el algoritmo **MCC** no da resultados precisos. En este caso está dando confiabilidad 1 (y varianza 0) para una red que no es perfecta. Esto se debe a que en todas las replicaciones se obtuvieron redes operativas. El algoritmo **RVR**, da un valor muy cercano a 1 con una pequeña varianza.

Comparación con resultados obtenidos en trabajos anteriores

En la siguiente tabla se muestran los resultados obtenidos (para la anti-confiabilidad) en [10] mediante los algoritmos de Ahmad (algoritmo exacto, [20]) y Antitético Generalizado ([10]) con 10^5, 10^6, 10^7 replicaciones respectivamente para las tres pruebas, contrastados con los resultados obtenidos mediante **RVR** con igual número de replicaciones.

	Q_V (Ahmad)	Q_V (Antitético)	**Var**$_{\mathbf{Antitético}}$	Q_V (RVR)	**Var**$_{\mathbf{RVR}}$
$p_n = 1$ $p_e = 0,9$	1,025E-01	1,046E-01	5,2E-07	1,018E-01	5,5E-07
$p_n = 1$ $p_e = 0,99$	1,094E-03	1,150E-03	1,1E-09	1,085E-03	6,9E-10
$p_n = 1$ $p_e = 0,999$	1,099E-05	1,198E-05	1,3E-12	1,081E-05	6,8E-13

En [10] no se cuenta con valores para la varianza del método Antitético Generalizado, pero en cambio sí se cuenta con los cocientes $Var_{MCC}/Var_{Antitético}$. En base a estos datos se estimó $Var_{Antitético}$ como R*(1-R) / (x*N) donde x= $Var_{MCC}/Var_{Antitético}$ y N es la cantidad de replicaciones (se utiliza el valor exacto de confiabilidad).

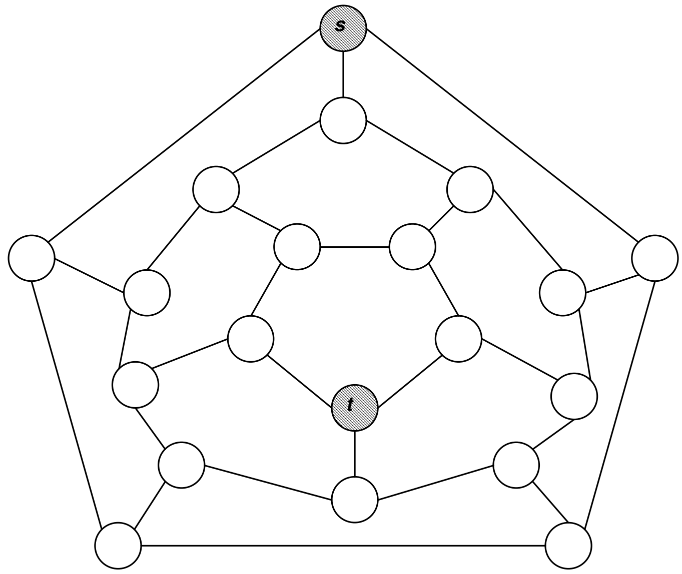
En general los resultados obtenidos por **RVR** y por el método Antitético Generalizado no difieren demasiado y son coherentes con los resultados exactos (ver que los resultados de **RVR** tienen más cifras en común con el método exacto y menor varianza para redes más confiables). En la siguiente tabla se comparan los errores relativos para los tres cálculos anteriores, donde se puede observar que en redes confiables son menores para **RVR**.

Antitético	RVR
6,9E-03	7,3E-03
2,9E-02	2,4E-02
9,5E-02	7,6E-02

Comparación de errores relativos en el cálculo de Q_V para RVR y Antitético Generalizado

Caso 4: Dodecaedro

Topología de red encontrada en varios trabajos anteriores como caso de prueba (20 nodos y 25 aristas). En [10] se cuenta con resultados numéricos para la medida $\boldsymbol{R_V}$ y en [8] para $\boldsymbol{R_{st}}$. El objetivo de este caso es realizar cálculos para las dos medidas de interés, comparar con resultados de trabajos anteriores y comparar errores relativos y eficiencias relativas para **MCC** y **RVR**. Se realizaron 36 pruebas con el mismo esquema del caso anterior.



Resultados (para R_v)

Prueba	MCC			RVR			Var_{MCC} /	T_{MCC} /	W_{MCC} /
	R_V	Var_{MCC}	T_{MCC}	R_V	Var_{RVR}	T_{RVR}	Var_{RVR}	T_{RVR}	W_{RVR}
p_n=0,9 p_e=0,9	0,1190	1,0E-05	4,7	0,1186976501	2,5E-08	71,5	4,3E+02	6,6E-02	2,8E+01
p_n=0,95 p_e=0,9	0,3509	2,3E-05	4,7	0,3499967713	2,1E-07	71,5	1,1E+02	6,6E-02	7,1E+00
p_n=0,98 p_e=0,9	0,6535	2,3E-05	4,7	0,6517986343	7,4E-07	71,5	3,0E+01	6,6E-02	2,0E+00
p_n=0,99 p_e=0,9	0,8006	1,6E-05	4,7	0,7985384349	1,1E-06	71,5	1,4E+01	6,6E-02	9,4E-01
p_n=0,999 p_e=0,9	0,9594	3,9E-06	4,7	0,9569774336	1,6E-06	71,5	2,4E+00	6,6E-02	1,6E-01
p_n=1,0 p_e=0,9	0,9788	2,1E-06	4,7	0,9763194297	1,7E-06	71,5	1,2E+00	6,6E-02	8,2E-02
p_n=0,9 p_e=0,95	0,1213	1,1E-05	4,7	0,1212397560	3,0E-09	71,7	3,6E+03	6,6E-02	2,3E+02
p_n=0,95 p_e=0,95	0,3578	2,3E-05	4,7	0,3574925293	2,6E-08	71,7	8,8E+02	6,6E-02	5,8E+01
p_n=0,99 p_e=0,95	0,6663	2,2E-05	4,7	0,6657579768	9,0E-08	71,8	2,5E+02	6,6E-02	1,6E+01
p_n=0,99 p_e=0,95	0,8163	1,5E-05	4,7	0,8156404523	1,4E-07	71,8	1,1E+02	6,6E-02	7,3E+00
p_n=0,999 p_e=0,95	0,9782	2,1E-06	4,7	0,9774726834	1,9E-07	71,8	1,1E+01	6,5E-02	7,2E-01
p_n=1,0 p_e=0,95	0,9980	2,0E-07	4,7	0,9972289203	2,0E-07	71,7	9,9E-01	6,6E-02	6,5E-02
p_n=0,9 p_e=0,98	0,1215	1,1E-05	4,7	0,1215486071	3,0E-10	71,8	3,6E+04	6,6E-02	2,4E+03
p_n=0,95 p_e=0,98	0,3584	2,3E-05	4,7	0,3584032203	2,6E-09	71,8	8,9E+03	6,6E-02	5,9E+02
p_n=0,98 p_e=0,98	0,6674	2,2E-05	4,7	0,6674539557	8,9E-09	71,8	2,5E+03	6,6E-02	1,6E+02
p_n=0,99 p_e=0,98	0,8177	1,5E-05	4,7	0,8177182479	1,3E-08	71,8	1,1E+03	6,6E-02	7,3E+01
p_n=0,999 p_e=0,98	0,9799	2,0E-06	4,7	0,9799627370	1,9E-08	71,8	1,0E+02	6,6E-02	6,8E+00
p_n=1,0 p_e=0,98	0,9997	3,0E-08	4,7	0,9997693017	2,0E-08	71,8	1,5E+00	6,5E-02	9,8E-02
p_n=0,9 p_e=0,99	0,1216	1,1E-05	4,7	0,1215761635	3,0E-14	71,9	3,6E+08	6,6E-02	2,4E+07
p_n=0,95 p_e=0,99	0,3585	2,3E-05	4,7	0,3584844743	2,6E-13	71,8	8,9E+07	6,6E-02	5,8E+06
p_n=0,98 p_e=0,99	0,6675	2,2E-05	4,7	0,6676052749	9,0E-13	71,8	2,5E+07	6,5E-02	1,6E+06
p_n=0,99 p_e=0,99	0,8178	1,5E-05	4,7	0,8179036336	1,4E-12	71,8	1,1E+07	6,6E-02	7,3E+05
p_n=0,999 p_e=0,99	0,9801	2,0E-06	4,7	0,9801849053	1,9E-12	71,8	1,0E+06	6,6E-02	6,6E+04
p_n=1,0 p_e=0,99	0,9999	1,0E-08	4,7	0,9999959605	2,0E-12	71,8	5,0E+03	6,5E-02	3,2E+02
p_n=0,9 p_e=0,999	0,1216	1,1E-05	4,7	0,1215766543	3,5E-19	71,8	3,1E+13	6,6E-02	2,0E+12
p_n=0,95 p_e=0,999	0,3585	2,3E-05	4,7	0,3584859217	3,0E-18	71,8	7,6E+12	6,6E-02	5,0E+11
p_n=0,98 p_e=0,999	0,6676	2,2E-05	4,7	0,6676079704	1,1E-17	71,8	2,1E+12	6,6E-02	1,4E+11
p_n=0,99 p_e=0,999	0,8179	1,5E-05	4,7	0,8179069360	1,6E-17	71,8	9,4E+11	6,5E-02	6,2E+10
p_n=0,999 p_e=0,999	0,9802	1,9E-06	4,7	0,9801888629	2,3E-17	71,7	8,5E+10	6,6E-02	5,6E+09
p_n=1,0 p_e=0,999	1	*	4,7	0,9999999980	2,4E-17	71,8	**	6,6E-02	**
p_n=0,9 p_e=1,0	0,1216	1,1E-05	4,7	0,1215766546	*	11,3	**	4,2E-01	**
p_n=0,95 p_e=1,0	0,3585	2,3E-05	4,7	0,3584859224	*	11,3	**	4,2E-01	**
p_n=0,98 p_e=1,0	0,6676	2,2E-05	4,7	0,6676079718	*	11,3	**	4,2E-01	**
p_n=0,99 p_e=1,0	0,8179	1,5E-05	4,8	0,8179069376	*	11,3	**	4,2E-01	**
p_n=0,999 p_e=1,0	0,9802	1,9E-06	4,7	0,9801888648	*	11,3	**	4,2E-01	**
p_n=1,0 p_e=1,0	1	*	4,7	1	*	11,3	**	4,2E-01	**

Los casilleros marcados con * corresponden a valores nulos obtenidos para la varianza.
En los casilleros marcados con ** no tiene sentido calcular los cocientes al contarse con numeradores o denominadores nulos.

Conclusiones

En la siguiente tabla se muestra una comparación del error relativo del estimador $\boldsymbol{Q_{est}}$ para la anti-confiabilidad ($\boldsymbol{Q_V}$), calculado en base a la desviación estándar y la esperanza, como:

$$\frac{\sigma_{Q_{est}}}{E(Q_{est})} = \frac{\sqrt{Var(Q_{est})}}{E(Q_{est})}$$

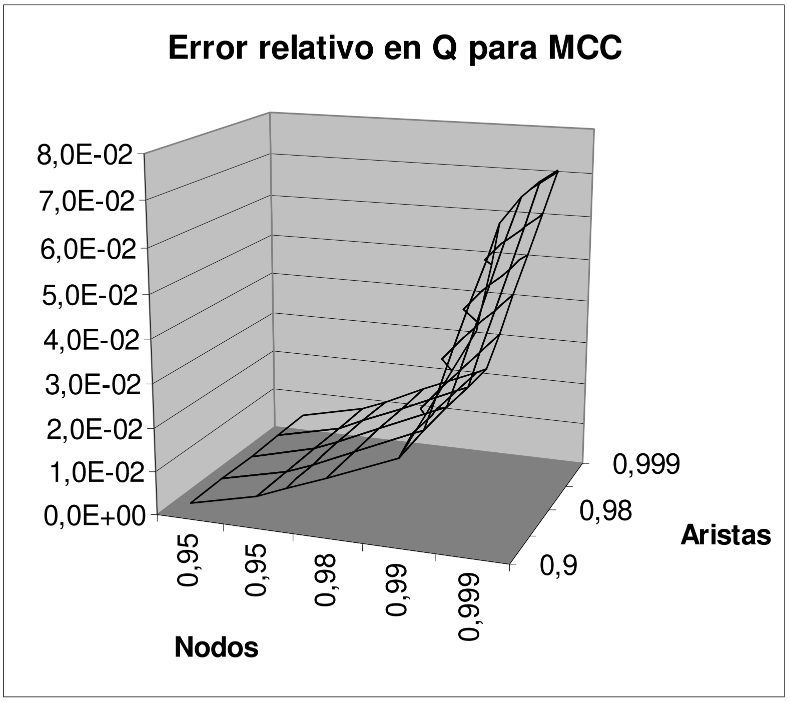
fijando nodos perfectos por un lado y aristas perfectas por otro.

	MCC					RVR				
	0,9	**0,95**	**0,98**	**0,99**	**0,999**	**0,9**	**0,95**	**0,98**	**0,99**	**0,999**
$\boldsymbol{p_n}$**=1**	6,8E-02	2,2E-01	5,8E-01	1,0E+00	*	5,5E-02	1,6E-01	6,1E-01	3,5E-01	2,4E+00
$\boldsymbol{p_e}$**=1**	3,7E-03	7,5E-03	1,4E-02	2,1E-02	7,0E-02	**	**	**	**	**

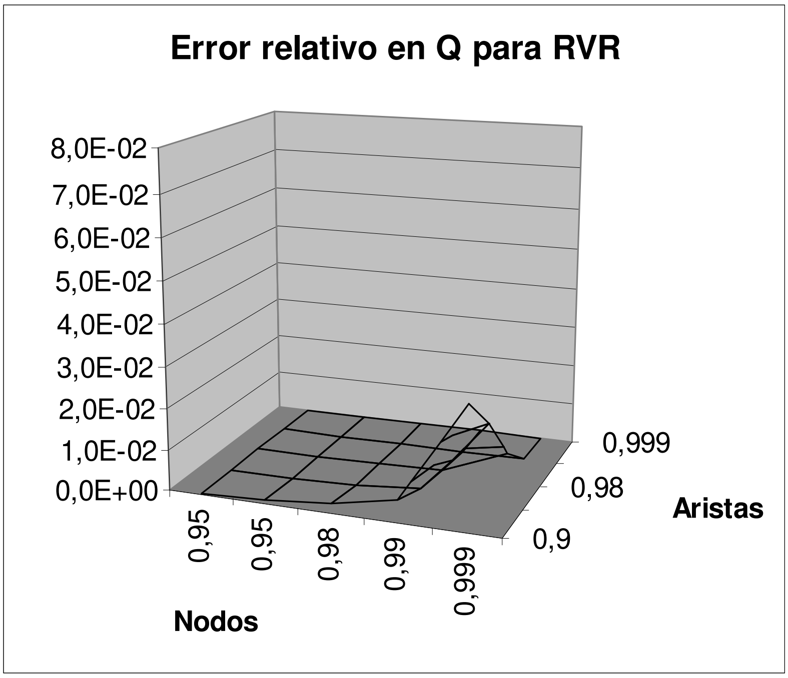
Los valores de la segunda fila corresponden a confiabilidades de aristas cuando se consideran nodos perfectos (tercer fila) y a confiabilidades de nodos cuando las aristas son perfectas (cuarta fila).
Los casilleros marcados con * corresponden a divisiones por cero (donde se obtuvieron valores de confiabilidad iguales a 1).
En los casilleros marcados con ** no es posible el cálculo del error ya que se obtuvieron valores nulos para la varianza.

- En general se puede ver que el error en **RVR** es menor que en **MCC**.
- También se puede observar que el error es mayor cuando se tienen nodos perfectos y varían las confiabilidades de las aristas (esta observación es válida solo para **MCC**).

El siguiente gráfico muestra los errores relativos en $\boldsymbol{Q_V}$ considerando fallas en nodos y aristas.



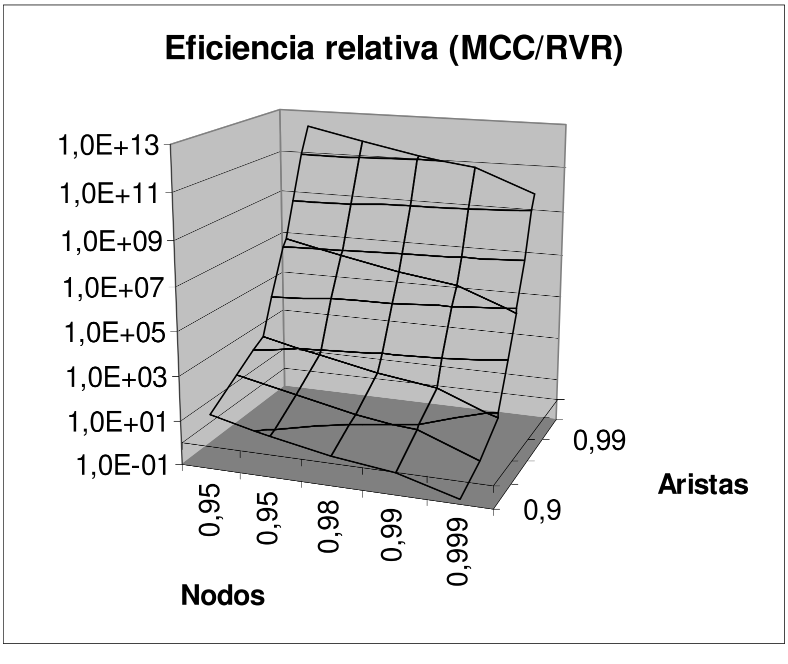
Error relativo en $\boldsymbol{Q_V}$ para MCC



Error relativo en Q_V para RVR

- Se puede observar que para **RVR** existen menos zonas donde los errores son altos. En particular estos se dan para confiabilidades medio-bajas para aristas y medio-altas para nodos, mientras que para **MCC** se dan para confiabilidades altas en nodos y cualquier confiabilidad en aristas.
- También se puede ver que los errores para **RVR** (máximo igual a 3,0E-02) son menores que para **MCC** (máximo igual a 7,0E-02).

El siguiente gráfico muestra la eficiencia relativa del algoritmo **MCC** con respecto a **RVR** para Q_V:



Eficiencia relativa (MCC/RVR) para Q_V en escala logarítmica

Se puede observar que para confiabilidades bajas en nodos y altas en aristas el trabajo de **MCC** es mayor, mientras que para confiabilidades altas en nodos y bajas en aristas el trabajo es mayor en **RVR** y en general para redes muy confiables es mayor el trabajo para **MCC**.

Comparación con resultados obtenidos en trabajos anteriores

En la siguiente tabla se muestran los resultados obtenidos (para la anti-confiabilidad) en [10] mediante los algoritmos de Ahmad (algoritmo exacto, [20]) y Antitético Generalizado [10] con 10^5, 10^6, 10^7 replicaciones respectivamente para las tres pruebas, contrastados con los resultados obtenidos mediante **RVR** con igual número de replicaciones.

	Q_V (Ahmad)	Q_V (Antitético)	$Var_{Antitético}$	Q_V (RVR)	Var_{RVR}
$p_n = 1 \quad p_e = 0,9$	2,287E-02	2,273E-02	3,7E-07	2,350E-02	1,6E-07
$p_n = 1 \quad p_e = 0,99$	2,030E-05	1,901E-05	1,8E-11	1,692E-05	1,2E-11
$p_n = 1 \quad p_e = 0,999$	2,003E-08	0	*	5,705E-09	-7,1E-18 **

En el casillero marcado con * no se cuenta en el trabajo original con el valor requerido.
El casillero marcado con ** presenta un valor negativo para la varianza (esto es debido a un error de precisión en los cálculos -ver implementación del cálculo de la varianza en **RVR**-).
Las varianzas del método Antitético Generalizado se calcularon de la misma forma que para el caso de prueba anterior.

Se pueden hacer las mismas observaciones que para el caso anterior. En la siguiente tabla se comparan los errores relativos para los dos primeros de los tres cálculos anteriores, donde se pueden observar valores del mismo orden de magnitud pero levemente menores para **RVR**.

Antitético	RVR
2,7E-02	1,7E-02
2,2E-01	2,0E-01

Comparación de errores relativos en el cálculo de Q_V para RVR y Antitético Generalizado (dos primeras pruebas)

Resultados (para R_{st})

Prueba	MCC			RVR			Var_{MCC} / Var_{RVR}	T_{MCC} / T_{RVR}	W_{MCC} / W_{RVR}
	R_{st}	Var_{MCC}	T_{MCC}	R_{st}	Var_{RVR}	T_{RVR}			
p_n=0,9 p_e=0,9	0,7860	1,7E-05	4,7	0,785171382	7,2E-07	134,4	2,3E+01	3,5E-02	8,2E-01
p_n=0,95 p_e=0,9	0,8938	9,5E-06	4,9	0,892733346	3,5E-07	139,3	2,7E+01	3,5E-02	9,5E-01
p_n=0,98 p_e=0,9	0,9556	4,2E-06	5,0	0,955091812	2,4E-07	142,5	1,7E+01	3,5E-02	6,1E-01
p_n=0,99 p_e=0,9	0,9763	2,3E-06	5,0	0,976296069	1,7E-07	143,5	1,3E+01	3,5E-02	4,7E-01
p_n=0,999 p_e=0,9	0,9954	4,6E-07	5,1	0,995592061	1,0E-07	144,4	4,6E+00	3,5E-02	1,6E-01
p_n=1,0 p_e=0,9	0,9975	2,5E-07	5,1	0,997122299	1,1E-09	78,2	2,3E+02	6,5E-02	1,5E+01
p_n=0,9 p_e=0,95	0,8019	1,6E-05	4,8	0,801319439	1,9E-07	135,6	8,4E+01	3,5E-02	3,0E+00
p_n=0,95 p_e=0,95	0,9010	8,9E-06	5,0	0,899870477	7,4E-08	140,4	1,2E+02	3,5E-02	4,3E+00
p_n=0,99 p_e=0,95	0,9599	3,8E-06	5,1	0,959879591	7,1E-10	143,1	5,4E+03	3,5E-02	1,9E+02
p_n=0,99 p_e=0,95	0,9797	2,0E-06	5,1	0,979704327	9,8E-09	144,0	2,0E+02	3,5E-02	7,1E+00
p_n=0,999 p_e=0,95	0,9978	2,2E-07	5,1	0,997641530	2,0E-08	145,4	1,1E+01	3,5E-02	3,9E-01
p_n=1,0 p_e=0,95	0,9998	2,0E-08	5,2	0,999704693	3,2E-11	79,2	6,2E+02	6,5E-02	4,1E+01
p_n=0,9 p_e=0,98	0,8064	1,6E-05	4,7	0,805606721	7,2E-08	136,4	2,2E+02	3,5E-02	7,5E+00
p_n=0,95 p_e=0,98	0,9020	8,8E-06	5,0	0,901805925	1,1E-08	140,7	8,3E+02	3,5E-02	2,9E+01
p_n=0,98 p_e=0,98	0,9603	3,8E-06	5,1	0,960303887	1,4E-10	142,9	2,7E+04	3,5E-02	9,7E+02
p_n=0,99 p_e=0,98	0,9800	2,0E-06	5,1	0,980063089	2,1E-11	143,6	9,4E+04	3,5E-02	3,3E+03
p_n=0,999 p_e=0,98	0,9980	2,0E-07	5,1	0,997991215	7,6E-14	144,6	2,6E+06	3,5E-02	9,3E+04
p_n=1,0 p_e=0,98	1	*	5,1	0,999983212	1,1E-14	78,4	**	6,5E-02	**
p_n=0,9 p_e=0,99	0,8071	1,6E-05	4,8	0,806559151	4,2E-08	136,6	3,7E+02	3,5E-02	1,3E+01
p_n=0,95 p_e=0,99	0,9022	8,8E-06	4,9	0,902070187	2,1E-09	140,3	4,3E+03	3,5E-02	1,5E+02
p_n=0,98 p_e=0,99	0,9604	3,8E-06	5,1	0,960347411	9,3E-11	143,0	4,1E+04	3,5E-02	1,5E+03
p_n=0,99 p_e=0,99	0,9801	2,0E-06	5,1	0,980089338	6,8E-12	144,0	2,9E+05	3,5E-02	1,0E+04
p_n=0,999 p_e=0,99	0,9980	2,0E-07	5,1	0,997999720	2,0E-15	144,6	1,0E+08	3,5E-02	3,6E+06
p_n=1,0 p_e=0,99	1	*	5,1	0,999997960	1,1E-16	78,4	**	6,5E-02	**
p_n=0,9 p_e=0,999	0,8079	1,6E-05	4,8	0,807571210	2,5E-08	136,4	6,2E+02	3,5E-02	2,2E+01
p_n=0,95 p_e=0,999	0,9023	8,8E-06	5,0	0,902150686	1,8E-09	140,4	4,9E+03	3,5E-02	1,7E+02
p_n=0,98 p_e=0,999	0,9604	3,8E-06	5,1	0,960383423	3,1E-11	142,9	1,2E+05	3,5E-02	4,4E+03
p_n=0,99 p_e=0,999	0,9801	2,0E-06	5,1	0,980096548	3,5E-12	143,6	5,6E+05	3,5E-02	2,0E+04
p_n=0,999 p_e=0,999	0,9980	2,0E-07	5,1	0,998000995	3,3E-17	144,4	6,0E+09	3,5E-02	2,1E+08
p_n=1,0 p_e=0,999	1	*	5,1	0,999999998	2,4E-17	78,3	**	6,5E-02	**
p_n=0,9 p_e=1,0	0,8079	1,6E-05	4,8	0,807337312	1,3E-08	50,4	1,2E+03	9,5E-02	1,2E+02
p_n=0,95 p_e=1,0	0,9023	8,8E-06	4,9	0,902186182	8,5E-10	51,9	1,0E+04	9,5E-02	9,8E+02
p_n=0,98 p_e=1,0	0,9604	3,8E-06	5,1	0,960383946	1,5E-11	52,8	2,5E+05	9,6E-02	2,4E+04
p_n=0,99 p_e=1,0	0,9801	2,0E-06	5,1	0,980097999	9,8E-13	52,9	2,0E+06	9,6E-02	1,9E+05
p_n=0,999 p_e=1,0	0,9980	2,0E-07	5,1	0,998000999	2,1E-18	53,1	9,6E+10	9,6E-02	9,2E+09
p_n=1,0 p_e=1,0	1	*	5,1	1	*	12,3	**	4,1E-01	**

Los casilleros marcados con * corresponden a valores nulos obtenidos para la varianza.
En los casilleros marcados con ** no tiene sentido calcular los cocientes al contarse con numeradores o denominadores nulos.

Conclusiones

Observando las tablas de resultados para $\boldsymbol{Q_v}$ y $\boldsymbol{Q_{st}}$ se puede ver que para esta última hay más casos donde la confiabilidad da igual a 1 para redes que no son perfectas.

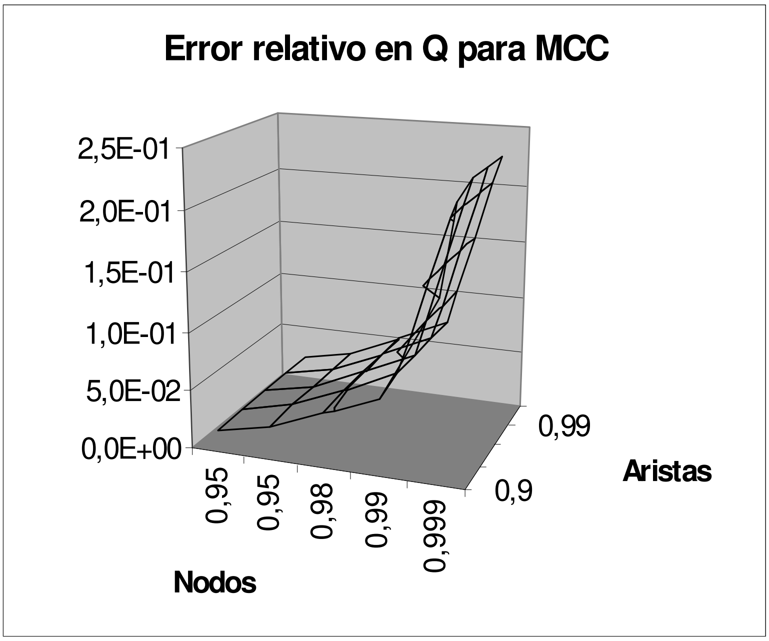
En la siguiente tabla se muestra una comparación del error relativo para la anti-confiabilidad ($\boldsymbol{Q_{st}}$) para nodos y aristas perfectas en forma análoga que para $\boldsymbol{Q_V}$.

	MCC					RVR				
	0,9	0,95	0,98	0,99	0,999	0,9	0,95	0,98	0,99	0,999
p_n=1	2,0E-01	7,1E-01	*	*	*	1,1E-02	1,9E-02	6,4E-03	5,3E-03	2,4E+00
p_e=1	2,1E-02	3,0E-02	4,9E-02	7,0E-02	2,2E-01	5,8E-04	3,0E-04	9,9E-05	5,0E-05	7,2E-07

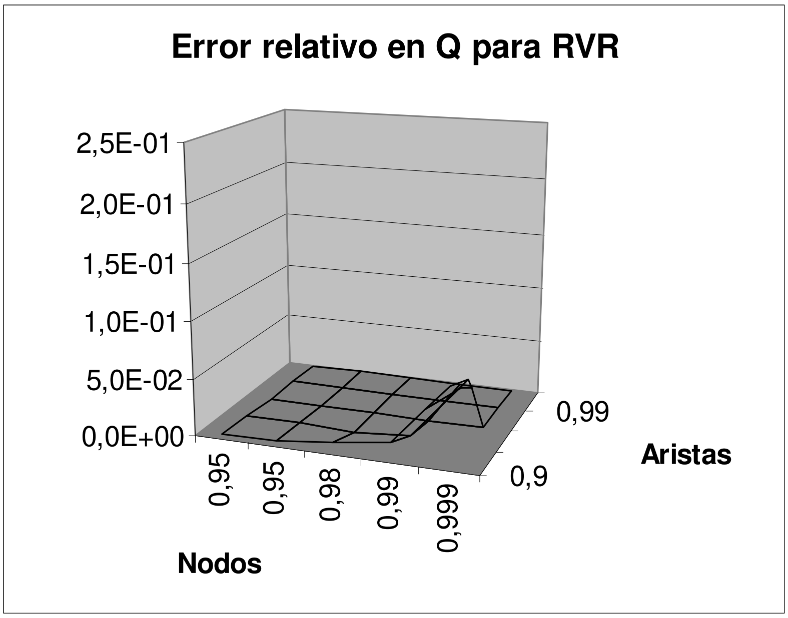
Los valores de la segunda fila corresponden a confiabilidades de aristas cuando se consideran nodos perfectos (tercer fila) y a confiabilidades de nodos cuando las aristas son perfectas (cuarta fila).
Los casilleros marcados con * corresponden a divisiones por cero (donde se obtuvieron valores de confiabilidad iguales a 1).

- Las mismas observaciones hechas para $\boldsymbol{Q_v}$ son válidas para $\boldsymbol{Q_{st}}$ con respecto a los errores relativos.
- También se puede ver que los errores cometidos en los cálculos de $\boldsymbol{Q_{st}}$ son en general mayores que los de $\boldsymbol{Q_v}$.

El siguiente gráfico muestra los errores relativos en $\boldsymbol{Q_{st}}$ considerando fallas en nodos y aristas.



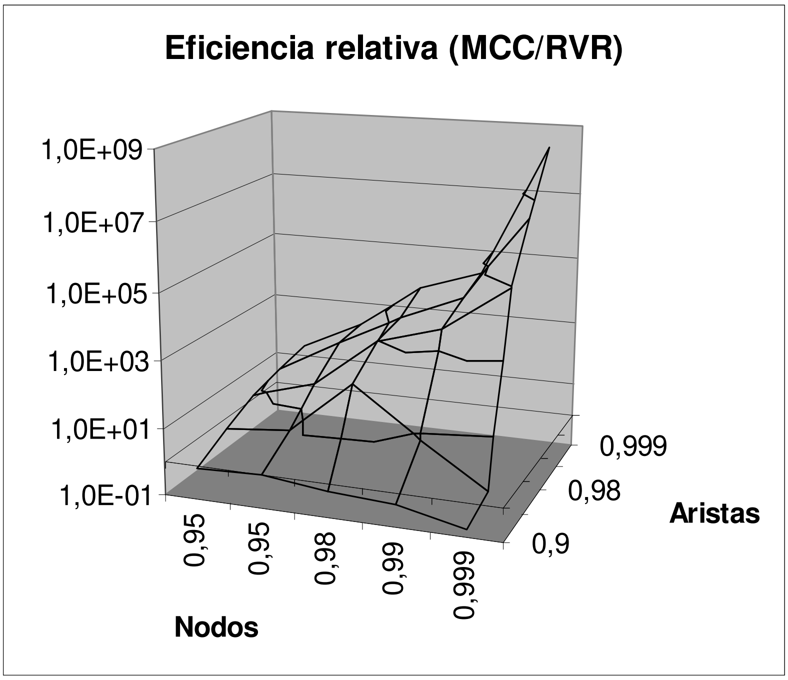
Error relativo en $\boldsymbol{Q_{st}}$ para MCC



Error relativo en $\boldsymbol{Q_{st}}$ para RVR

En general se observan las mismas zonas de precisión para el cálculo de $\boldsymbol{Q_{st}}$ que para $\boldsymbol{Q_v}$ en ambos algoritmos.

El siguiente gráfico muestra la eficiencia relativa del algoritmo **MCC** con respecto a **RVR** para Q_{st}:



Eficiencia relativa (MCC/RVR) para Q_{st} en escala logarítmica

Se puede observar que la zona donde el trabajo es mayor para **RVR** es la misma que para Q_V (nodos muy confiables y aristas poco confiables) mientras que cambian las zonas de mayor esfuerzo para **MCC** (observar que el pico corresponde a redes muy confiables lo que confirma la conveniencia de **RVR** para redes de este tipo).

Comparación con resultados obtenidos en trabajos anteriores

En la siguiente tabla se muestran los resultados obtenidos (para la anti-confiabilidad) en [8] mediante los algoritmos de factorización [13] (primer columna) y **RVR** (original) para R_{st} [8] (segunda columna) con 10^6 replicaciones, contrastándolos con los resultados obtenidos mediante el algoritmo **RVR** implementado en nuestro trabajo.

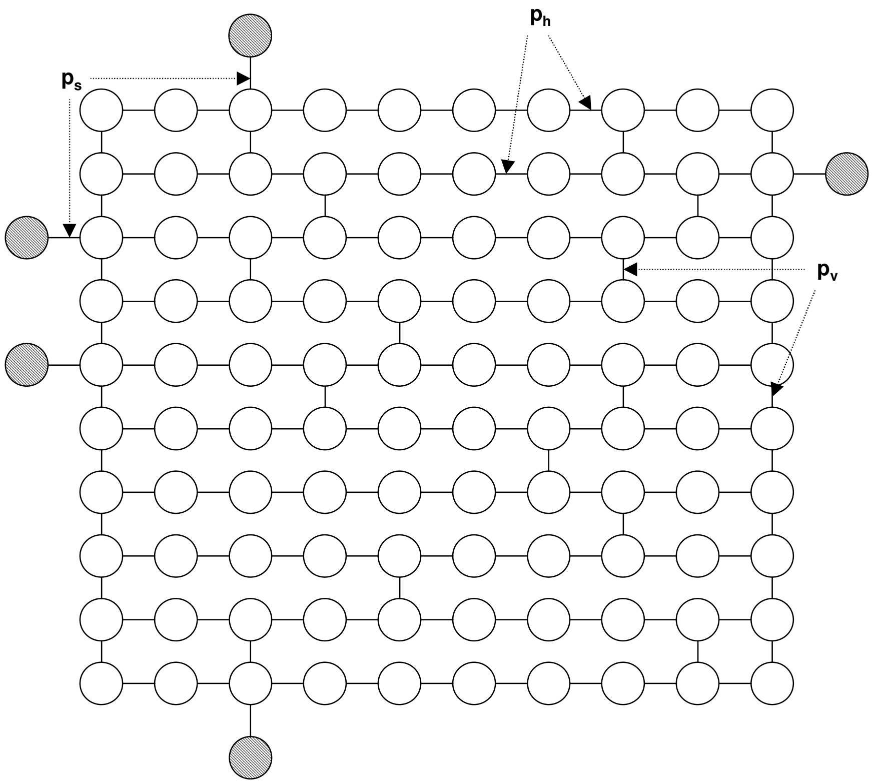
	Q_{st} (exacto)	Q_{st} (original)	**Var** (original)	Q_{st} (RVR)	Var_{RVR}
$p_n = 1$ $p_e = 0,9$	2,880E-03	2,888E-03	8,99E-11	2,881E-03	1,2E-11
$p_n = 1$ $p_e = 0,95$	2,946E-04	2,937E-04	1,91E-12	2,945E-04	1,2E-13
$p_n = 1$ $p_e = 0,98$	1,702E-05	1,702E-05	9,80E-15	1,702E-05	6,1E-16
$p_n = 1$ $p_e = 0,99$	2,062E-06	2,062E-06	1,66E-16	2,058E-06	-1,2E-17**
$p_n = 1$ $p_e = 0,999$	2,006E-09	2,006E-09	1,79E-22	1,988E-09	-1,4E-17**

En los casilleros marcados con ** se obtuvieron valores negativos para la varianza (ver mismo problema para Q_V).

Se puede observar que para las tres primeras pruebas los resultados obtenidos en el trabajo original y en el nuestro coinciden (y también con los resultados exactos). Para las dos últimas pruebas (redes muy confiables) los resultados originales son más precisos. Esto puede deberse a la implementación del algoritmo, donde posiblemente en el trabajo original sea específica para Q_{st} mientras que en el nuestro se utiliza la misma implementación que para Q_V y Q_K, con un conjunto particular de terminales de cardinalidad 2.

Caso 5: Malla dispersa

Se considera una topología de red con forma de malla (10×10 nodos más 5 terminales y 127 aristas), tomado de [11] donde se cuenta con resultados numéricos. La medida de interés en este caso es $\boldsymbol{R_K}$. Los dos objetivos principales de este caso son realizar pruebas con topologías con confiabilidades dispares en sus componentes (se distinguen entre aristas horizontales, verticales y terminales), y comparar con los resultados obtenidos en el trabajo original.



Malla dispersa

En el trabajo original se muestran resultados para distintos valores de p_h, p_v y p_s. En nuestro trabajo se realizaron pruebas con los mismos valores considerados en [11].

	Prueba 1	Prueba 2	Prueba 3	Prueba 4
p_h	0,98	0,99	0,995	0,995
p_v	0,99	0,99	0,99	0,99
p_s	0,999	0,999	0,999	0,9995

Se realizaron en total 8 pruebas (4 con nodos perfectos y 4 con confiabilidad 0,99 en los mismos).

Resultados

Los resultados se muestran de acuerdo al cuadro presentado anteriormente en la descripción del caso de prueba. Las 4 primeras pruebas corresponden a confiabilidades 0,99 en nodos y las 4 últimas, a redes con nodos perfectos.

p_n	Prueba	MCC			RVR			Var_{MCC}/ Var_{RVR}	T_{MCC}/ T_{RVR}	W_{MCC}/ W_{RVR}
		R	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
0,99	1	0,8977	9,2E-06	80,6	0,8967	3,91E-06	9863,2	2,3E+00	8,2E-03	1,9E-02
	2	0,8986	9,1E-06	80,2	0,8974	3,87E-06	9910,0	2,4E+00	8,1E-03	1,9E-02
	3	0,8986	9,1E-06	80,2	0,8983	3,79E-06	9911,5	2,4E+00	8,1E-03	1,9E-02
	4	0,9005	9,0E-06	80,1	0,9000	3,68E-06	9916,5	2,4E+00	8,1E-03	2,0E-02
1,0	5	0,9961	3,9E-07	81,4	0,9942	3,67E-07	5629,6	1,1E+00	1,4E-02	1,5E-02
	6	0,9962	3,8E-07	81,3	0,9949	3,08E-07	5633,5	1,2E+00	1,4E-02	1,8E-02
	7	0,9962	3,8E-07	81,4	0,9946	3,38E-07	5635,1	1,1E+00	1,4E-02	1,6E-02
	8	0,9983	1,7E-07	81,3	0,9980	9,97E-08	5654,3	1,7E+00	1,4E-02	2,4E-02

Conclusiones

- Se puede ver que el algoritmo **MCC** da prácticamente los mismos valores para cambios sensibles en las confiabilidades de las componentes de la red (no es el caso de **RVR**).
- Las medidas de las eficiencias relativas muestran en todos los casos mayor trabajo para **RVR**. Esto se debe al tamaño relativamente grande de la red que hace que los tiempos de ejecución sean tan altos que no compensen la reducción de la varianza.

Comparación con resultados obtenidos en trabajos anteriores

Los resultados obtenidos en el trabajo original mediante el algoritmo de Destrucción Secuencial (**DS**) son los siguientes (para nodos perfectos):

Prueba	R_{RVR}	Var_{RVR}	R_{DS}	Var_{DS}
5	0,9942	3,7E-07	0,956	1,5E-06
6	0,9949	3,1E-07	0,985	2,6E-07
7	0,9946	3,4E-07	0,9923	8,2E-08
8	0,9980	1,0E-07	0,9948	4,2E-08

En [11] no se cuenta con valores para la varianza, pero en cambio sí se cuenta con los cocientes Var_{MCC}/Var_{DS}. En base a estos datos se estimó Var_{DS} como R*(1-R) / (x*N) donde x= Var_{MCC}/Var_{DS} y N es la cantidad de replicaciones (se utiliza el único valor de confiabilidad disponible que es el calculado mediante DS).

Se observan diferencias importantes entre los resultados obtenidos en el trabajo original (tercer columna de la tabla anterior) y los obtenidos en el nuestro (primer columna). Los intervalos de confianza para los dos métodos son disjuntos en los dos primeros casos (pruebas 5 y 6).

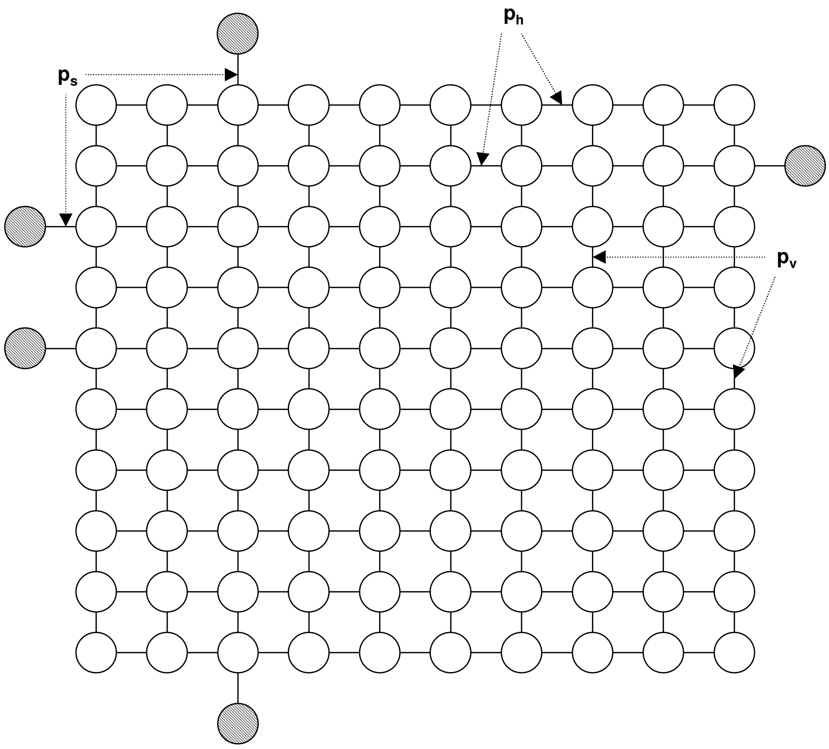
Se realizó una prueba adicional que aportara más información para la validación de nuestros resultados. Se consideraron dos pruebas para el caso 6 (p_v=p_h=0,99, p_s=0,999 y nodos perfectos): la primera (red 1) con p_s=1 y p_v=p_h=0,99 y la segunda (red 2) con p_s=0,999 y p_v=p_h=1. La confiabilidad de la red original se puede calcular como el producto de las confiabilidades de las redes de las dos pruebas anteriores.

Los resultados obtenidos son 0,999959 para la red 1 y 0,995006 para la red 2 y el producto es 0,994965.

Esto contribuye a validar los resultados obtenidos en nuestras pruebas. También puede observarse que los resultados obtenidos mediante los dos algoritmos (**MCC** y **RVR**) son relativamente parecidos. Las diferencias con los resultados originales pueden justificarse por una interpretación incorrecta del artículo, errores en la escritura del mismo o incluso en la implementación de los algoritmos originales.

Caso 6: Malla densa

Topología similar a la anterior, pero con 90 aristas verticales (10×10 nodos más 5 terminales y 180 aristas). La medida de interés también es R_K (siendo K el mismo conjunto de terminales que para el caso anterior). El objetivo de este caso de prueba es observar el aumento de la confiabilidad de la red al aumentar la cantidad de aristas y también el aumento de los tiempos de ejecución.



Malla densa

Se considera el mismo conjunto de pruebas que para el caso anterior.

Resultados y conclusiones

Los resultados se presentan (en la siguiente página) en el mismo formato que en el caso anterior.

- Se puede observar que en general los tiempos de ejecución no aumentaron demasiado (insignificante para **MCC** y del orden del 2% para **RVR**).
- Las confiabilidades aumentaron en algunos casos y en otros disminuyeron, pero se debe tener en cuenta también las varianzas y no solos las medias. En particular se puede observar que los valores de las medias siempre aumentan al aumentar las confiabilidades de los componentes (ver resultados para todos los casos de prueba anteriores), pero no al aumentar la densidad de la red (si bien teóricamente el valor exacto de la confiabilidad de la red debería aumentar). Esto se puede justificar en los casos de R_k y en particular R_{st}, donde la inserción de una nueva arista entre un par de vértices no adyacentes no siempre contribuye en formar caminos alternativos a los ya existentes entre terminales.

En la siguiente página se muestran los resultados para este caso de prueba y se presentan nuevamente los resultados para el caso anterior, donde se puede observar en las filas resaltadas (casos donde las medias para la malla densa dieron menores que para la malla dispersa) que los intervalos de confianza tienen intersección no vacía.

P_n	Prueba	MCC			RVR			Var_{MCC}/ Var_{RVR}	T_{MCC}/ T_{RVR}	W_{MCC}/ W_{RVR}
		R	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
0,99	1	0,8997	9,0E-06	82,4	0,9006	3,6E-06	10178,4	2,5E+00	8,1E-03	2,0E-02
	2	0,8998	9,0E-06	82,3	0,9014	3,5E-06	10162,9	2,6E+00	8,1E-03	2,1E-02
	3	0,8998	9,0E-06	82,5	0,8980	3,8E-06	10184,1	2,4E+00	8,1E-03	1,9E-02
	4	0,9020	8,8E-06	82,2	0,9029	3,4E-06	10181,3	2,6E+00	8,1E-03	2,1E-02
1,0	5	0,9952	4,8E-07	83,4	0,9946	3,4E-07	5746,7	1,4E+00	1,5E-02	2,1E-02
	6	0,9952	4,8E-07	83,4	0,9943	3,7E-07	5746,1	1,3E+00	1,5E-02	1,9E-02
	7	0,9952	4,8E-07	83,4	0,9944	3,6E-07	5743,6	1,3E+00	1,5E-02	1,9E-02
	8	0,9976	2,4E-07	83,5	0,9978	1,2E-07	5743,4	2,0E+00	1,5E-02	2,9E-02

Resultados para malla densa

p_n	Prueba	MCC			RVR			Var_{MCC}/ Var_{RVR}	T_{MCC}/ T_{RVR}	W_{MCC}/ W_{RVR}
		R	Var_{MCC}	T_{MCC}	*R*	Var_{RVR}	T_{RVR}			
0,99	1	0,8977	9,2E-06	80,6	0,8967	3,9E-06	9863,2	2,3E+00	8,2E-03	1,9E-02
	2	0,8986	9,1E-06	80,2	0,8974	3,9E-06	9910,0	2,4E+00	8,1E-03	1,9E-02
	3	0,8986	9,1E-06	80,2	0,8983	3,8E-06	9911,5	2,4E+00	8,1E-03	1,9E-02
	4	0,9005	9,0E-06	80,1	0,9000	3,7E-06	9916,5	2,4E+00	8,1E-03	2,0E-02
1,0	5	0,9961	3,9E-07	81,4	0,9942	3,7E-07	5629,6	1,1E+00	1,4E-02	1,5E-02
	6	0,9962	3,8E-07	81,3	0,9949	3,1E-07	5633,5	1,2E+00	1,4E-02	1,8E-02
	7	0,9962	3,8E-07	81,4	0,9946	3,4E-07	5635,1	1,1E+00	1,4E-02	1,6E-02
	8	0,9983	1,7E-07	81,3	0,9980	1,0E-07	5654,3	1,7E+00	1,4E-02	2,4E-02

Resultados para malla dispersa

Conclusiones generales

Las principales conclusiones generales extraídas de todas las pruebas realizadas son las siguientes:

- El algoritmo **GCE** se hace impracticable para redes de tamaño aproximado a 15 nodos y 19 aristas.
- Las varianzas de **RVR** son siempre menores que las de **MCC** y esta diferencia se acentúa al aumentar la confiabilidad de la red.
- Los tiempos de ejecución de **RVR** son siempre mayores que los de **MCC** y la diferencia aumenta al aumentar el tamaño de la red.
- Los resultados de confiabilidad en **MCC** son menos sensibles a los cambios en las confiabilidades de los componentes, que en **RVR**. Esto se puede justificar observando que al cambiar sensiblemente la confiabilidad de un componente, los sorteos que determinan el funcionamiento o no de un componente en **MCC** pueden dar el mismo resultado (que siempre será 0 o 1); este no es el caso de **RVR**, donde se tienen en cuenta más valores exactos (probabilidad de falla del corte elegido) en la estimación de la medida.
- Los errores relativos para ambos algoritmos, en una determinada topología, son mayores para ciertos valores de confiabilidades de nodos y aristas (ver caso de prueba 4 -Dodecaedro-). Las zonas de errores relativos son distintas para distintas medidas (Q_v y Q_{st} en el mismo ejemplo).
- En algunos casos se presentaron errores de precisión numérica, en particular en el cálculo de varianzas en **RVR** para redes muy confiables (ver la comparación con resultados obtenidos en trabajos anteriores, en el cálculo de la medida Q_v para el caso 4 -Dodecaedro-).

- En general los resultados obtenidos son coherentes con los obtenidos en trabajos anteriores, salvo para el caso 5 (malla dispersa) donde se observan diferencias importantes. En las comparaciones con resultados obtenidos en trabajos anteriores de las pruebas para dicho caso, se justifican las posibles causas de la discrepancia.
- Los resultados obtenidos para las confiabilidades del caso 6 (malla densa) muestran (en relación a los obtenidos para el caso 5 -malla dispersa-) que no siempre un aumento en la cantidad de componentes de la red (aristas en este caso) se ve reflejado en un aumento en las medias de confiabilidad obtenidas con los algoritmos estimativos. En estos casos se debe prestar especial atención a las varianzas y construir los intervalos de confianza.

6 - Integración con HEIDI

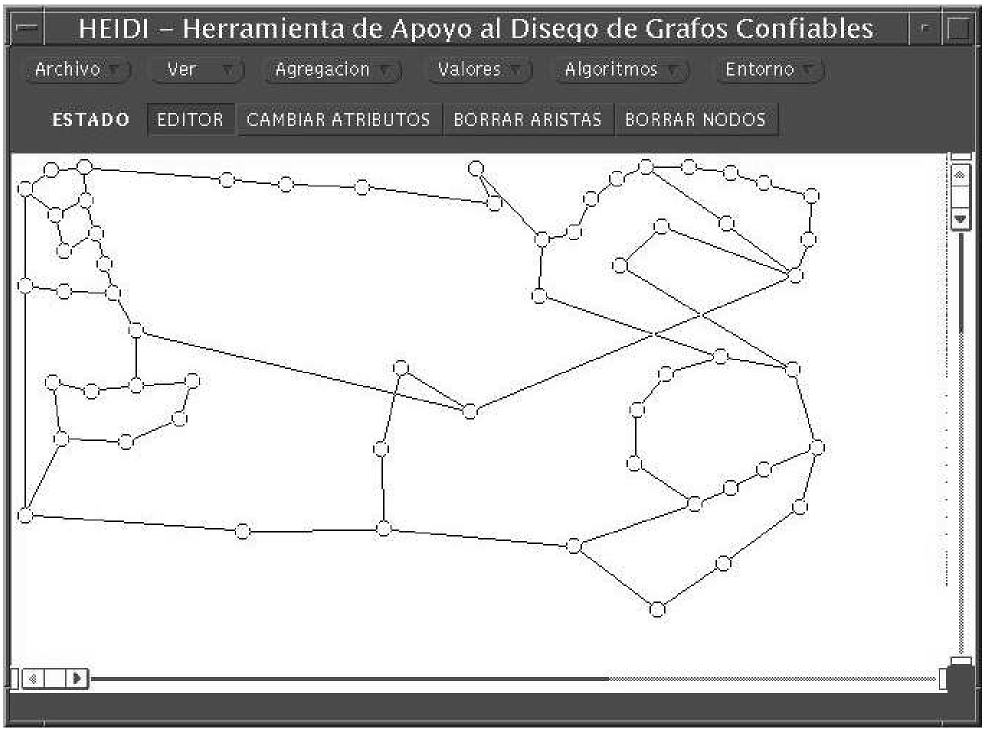
En este capítulo se presenta la parte de trabajo realizado que tuvo como objetivo la incorporación del algoritmo **RVR** implementado, a la herramienta HEIDI (Herramienta Inteligente para el Diseño de Redes de Comunicación Confiables, [9]). Se describen las principales características de la herramienta y los principales cambios realizados para lograr la integración.

Descripción de la herramienta y objetivos de la integración

La herramienta HEIDI fue desarrollada a través de varios proyectos de investigación y de grado, con el fin de contar con un software que permita testear y utilizar diversos algoritmos sobre redes (principalmente aquellos relacionados con el tema de confiabilidad), de forma sencilla. Las principales características de la misma son:

Desde el punto de vista del usuario:

- Interfase gráfica que permite dibujar la topología de la red utilizando el mouse.
- Posibilidad de trabajar con redes orientadas y no orientadas.
- Posibilidad de asignar distintos valores (capacidad, costo y confiabilidad) a nodos y aristas de forma sencilla.
- Cálculo de parámetros de redes como ser confiabilidad, vulnerabilidad, conectividad y otros.
- Algoritmos de optimización de redes.



Editor gráfico

Desde el punto de vista del programador:

- La herramienta fue desarrollada en el lenguaje C++ bajo ambiente Unix y en particular utiliza componentes del entorno gráfico Open Windows.
- Se cuenta con un diseño de la estructura de módulos adecuado para realizar extensiones.

En nuestro caso particular la extensión consiste en agregar un nuevo algoritmo para el cálculo de la confiabilidad de una red a los ya existentes. A grandes rasgos, los procedimientos necesarios para realizar la extensión son los siguientes:

1 - Agregar una nueva opción en el menú de algoritmos.
2 - Conectar la selección del nuevo algoritmo con la invocación al mismo.

Relevamiento de información

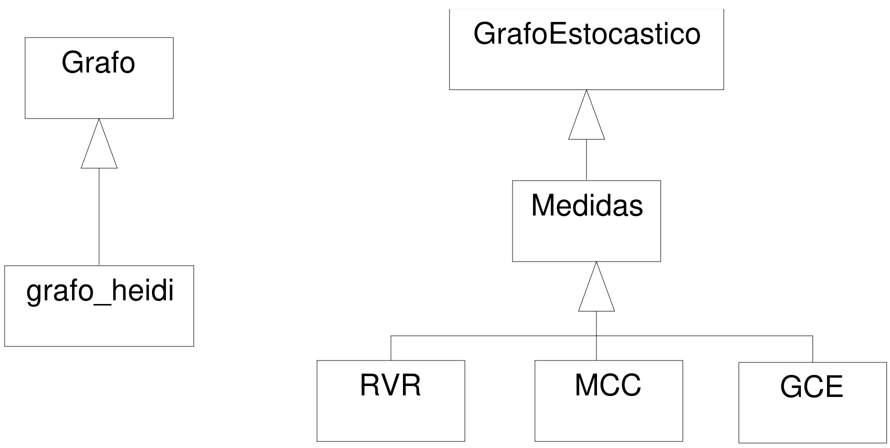
La tarea de extensión de la herramienta con un nuevo algoritmo requirió un estudio previo de las estructuras existentes para la representación de grafos (por parte del editor gráfico) y su conexión con la estructura utilizada en nuestro trabajo. La información necesaria se obtuvo de [9] y mediante inspección directa del código original (se cuenta con los fuentes de todos los módulos que conforman la aplicación, así como un *makefile* que funciona correctamente). Los puntos relevantes que se extrajeron son los siguientes:

- La información que se ingresa desde el editor se almacena en una estructura de clase **Grafo** que básicamente contiene la representación de la topología y los valores asignados a nodos y aristas.
- La estructura mencionada anteriormente soporta operaciones como inserción y borrado de nodos, lo que requiere de una implementación que no es eficiente para la ejecución de algoritmos sobre la misma.
- Para solucionar el problema anterior se cuenta con una clase denominada **grafo_heidi** que hereda de **Grafo** y realiza la transformación (mediante un constructor) de una estructura a otra. Esta clase posee una estructura de datos declarada `protected` para poder acceder a la misma desde cualquier clase derivada.

En nuestro trabajo, las clases principales desarrolladas (que implementan las estructuras de datos y los algoritmos) son las siguientes:

- **GrafoEstocastico**: Implementa el modelo matemático grafo, en particular no orientado y estocástico.
- **Medidas**: Hereda de **GrafoEstocastico** e implementa operaciones que son comunes a varios algoritmos de cálculo de confiabilidad sobre redes.
- **RVR**: Hereda de **Medidas** e implementa el cálculo de la confiabilidad de una red mediante el método de Reducción Recursiva de la Varianza (algoritmo **RVR**).

Análogamente se tienen las clases **MonteCarlo** y **GenCompleta** que implementan el cálculo de la confiabilidad mediante los métodos Monte Carlo crudo (algoritmo **MCC**) y generación completa de estados (algoritmo **GCE**) respectivamente.

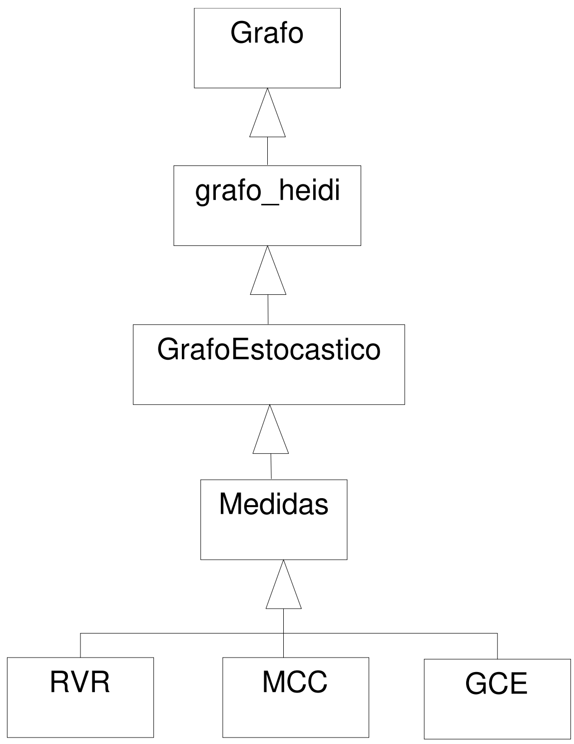


Clases de HEIDI y de nuestro trabajo

A continuación se explica el proceso de extensión para el cálculo mediante **RVR** (para los otros dos casos el proceso es análogo).

Procedimiento de extensión

El procedimiento para el pasaje de estructuras propuesto en [9] consiste en crear una nueva clase que contenga la nueva estructura, que herede de la clase **grafo_heidi** (para poder acceder a su estructura `protected`) con un constructor de la forma `NuevaClase::NuevaClase(grafo_heidi *)`.
En nuestro caso la nueva clase es **GrafoEstocastico** por lo que se debe agregar a la misma un constructor de la forma `GrafoEstocastico::GrafoEstocastico(grafo_heidi *)`.
Luego debe agregarse a la clase **RVR** un constructor de la forma `RVR::RVR(GrafoEstocastico *)` o `RVR::RVR(GrafoEstocastico &)`.



Integración de las clases

Las clases **GrafoEstocastico**, **Medidas**, **RVR** y todas las clases auxiliares se colocaron juntas en los archivos `todoRVR.h` y `todoRVR.cc` (definiciones e implementaciones respectivamente para cada clase).

Los archivos de HEIDI que deben modificarse para realizar la extensión son los siguientes:

- `nomfun.h`: Contiene entre otras cosas la definición de los nombres de las funciones que se despliegan en el menú de algoritmos.
- `funcs.cc`: Contiene la función que se invoca al seleccionar una opción del menú de algoritmos, que se encarga de invocar el algoritmo seleccionado. También contiene cada uno de los procedimientos que se invocan para cada algoritmo.
- `makefile`: Se deben agregar las nuevas reglas para compilación y linkedición.

Teniendo en cuenta el procedimiento completo indicado en [9] para realizar la extensión, los pasos que se siguieron son los siguientes:

1 - Crear un nuevo módulo con las nuevas estructuras y algoritmos (en nuestro caso son los archivos `todoRVR.h` y `todoRVR.cc`). En el archivo `todoRVR.h` se debe incluir la directiva `#include "grafo.h"` para poder utilizar la clase **grafo_heidi**.

2 - Agregar en el archivo `nomfun.h` un elemento al arreglo `char * nombres_calculos[]` con un nombre representativo para la nueva operación (es la nueva opción que desplegará el menú de algoritmos que en nuestro caso es "Confiab Rk (RVR)").

3 - Agregar en el archivo `funcs.cc` la función que atiende el evento producido por la selección en el menú de la nueva opción. Esta rutina debe realizar las siguientes acciones:

- Crear un objeto de clase **grafo_heidi** a partir del objeto global *grafo_main* de clase **Grafo**.
- Crear un objeto de la nueva clase (en nuestro caso es **RVR**) utilizando el constructor de la misma que recibe un objeto de clase **grafo_heidi**.
- Invocar el método de la nueva clase que realiza el nuevo cálculo (en nuestro caso es el método **conf** de la clase **RVR**).
- Invocar el procedimiento `desplegar_res(char *nombre_fun,char *res)` que recibe el nombre de la función (el mismo que se agregó al arreglo `nombres_calculos`) y un string conteniendo el resultado a desplegar.

Para el nombrado de la rutina se sigue la convención `c_nombre_calculo` por lo que se denominó `void c_conf_Rk_RVR()` (ver figura en siguiente página).

4 - Extender la función que atiende el menú de cálculos con el nuevo caso para invocar cuando corresponda, la nueva función definida en 3. Para el caso del cálculo de confiabilidad mediante **RVR** se necesita un parámetro extra que se obtiene en forma interactiva (número de replicaciones), para lo que se deben seguir los siguientes pasos:

- En el procedimiento `funciones_de_heidi_sp` (en el archivo `funcs.cc`) debe agregarse una nueva condición que corresponde a la selección del menú de opciones del nuevo nombre de función agregado, donde se debe asignar a la variable `listaparametros`, el string con la descripción del parámetro que debe ingresarse.
- En el procedimiento `funciones_de_heidi_cp` (en el archivo `funcs.cc`) debe agregarse una nueva condición (de forma análoga que para el caso anterior), donde se invocará la función definida en 3.

5 - En el archivo `makefile` debe agregarse la nueva regla para compilar el o los nuevos archivos (en nuestro caso es un sólo archivo denominado `todoRVR.cc`) y se debe modificar la regla para el linker, de forma de incluir el nuevo módulo.

Los pasos a seguir para la integración de los algoritmos **MCC** y **GCE** son análogos a los ya descritos (para **GCE** no se requieren parámetros de entrada).

```
void c_conf_Rk_RVR() {
  double tiempo, media, var;
  char buf[20];
  ResultMC resultado;

  grafo_heidi * graf_local = new grafo_heidi(grafo_main);
  // crea un grafo_heidi a partir de un Grafo
  GrafoEstocastico *ge = new GrafoEstocastico(*graf_local);
  // crea un GrafoEstocastico a partir de un grafo_heidi
  RVR *graf_local_RVR = new RVR(*ge);
  // crea un RVR a partir de un GrafoEstocastico
  resultado = graf_local_RVR->conf(1,fparametros[1],tiempo);
  // invocación a la función de cálculo de confiabilidad
  // (con el torrente número 1 de número seudo-aleatorios y
  // cantidad de replicaciones tomadas de los parámetros de entrada)
  media = resultado.darMedia();
  var = resultado.darVar();
  sprintf(buf,"C=%.5f/V=%.1E",media,var);
  desplegar_res("Confiab Rk (RVR)",buf);
  // despliega los resultados en la pantalla del editor
  delete ge;
  delete graf_local;
  delete graf_local_RVR;
}
```

Pasaje de estructura, invocación a la función de cálculo y presentación de resultados

Cambios en el código original

Para el correcto funcionamiento de los nuevos algoritmos (cálculo de la confiabilidad R_k mediante **RVR**, **MCC** y **GCE**) debieron realizarse los siguientes cambios en el código original de HEIDI:

- En el archivo `grafo.h`, en la clase **grafo_heidi** se agrega una línea que indica `friend GrafoEstocastico` para así poder acceder a su estructura `protected` desde esta última (el hecho de que **GrafoEstocastico** herede de **grafo_heidi** permite acceder a su estructura protegida pero solamente desde el objeto implícito de clase **GrafoEstocastico** y no desde un objeto de clase **grafo_heidi** pasado como parámetro, que es el caso del constructor). También se agrega antes de la declaración de la clase **grafo_heidi** una línea con la etiqueta `class GrafoEstocastico`, que permite que la clase **grafo_heidi** pueda reconocer la existencia de tal clase, implementada en otro módulo.

- En el archivo `grafo.cc`, en la clase **grafo_heidi**, se agrega en las operaciones `grafo_heidi::grafo_heidi(Grafo *)` y `grafo_heidi::grafo_heidi(const grafo_heidi &)` código necesario para copiar los valores de los nodos (costo, capacidad y confiabilidad). Esto no estaba considerado hasta el momento pues se trabajaba con modelos con fallas en aristas únicamente.

- En el archivo `makefile` se agregó una regla para la compilación del archivo `todoRVR.cc`, y se agregó el archivo `todoRVR.o` en la lista de módulos a linkeditarse.

Consideraciones de implementación

Los algoritmos disponibles en la herramienta para el cálculo de confiabilidad trabajan todos con la medida R_v (existencia de camino entre todo par de nodos). La dificultad de trabajar con R_k o

R_{st} radica en la necesidad de distinguir los nodos terminales desde la interfase del editor. Para la integración de nuestros algoritmos se tomó la convención de marcar dichos nodos con un valor particular para su costo (valor que no se utiliza para el cálculo de la confiabilidad). De esta forma, el usuario debe marcar los nodos terminales asignando valor 1 a su costo, y esta información se procesará al realizar la transformación de estructuras.

Los algoritmos **MCC** y **GCE** se integraron en forma análoga.

7 - Difusión de resultados

En este capítulo se describe la parte del trabajo realizado que tuvo como objetivo la difusión del trabajo, conclusiones y resultados obtenidos. Se describen a grandes rasgos las características y detalles de implementación del sitio web y el servidor de cálculo construidos.

El medio escogido para la difusión fue la construcción de un sitio web, y se agregó como objetivo a esta parte del proyecto, la construcción de una herramienta que permitiera a usuarios remotos el acceso a las ejecuciones de los algoritmos implementados. Para lograr dichos objetivos se implementaron las siguientes herramientas:

1 - **Sitio web**: Básicamente contiene un resumen del presente informe.
2 - **Servidor de cálculo**: Permite a un usuario remoto enviar su propio caso de prueba y obtener los resultados de los cálculos, efectuados con los algoritmos implementados en nuestro trabajo.

Sitio web

En el sitio web ubicado en la dirección http://www.fing.edu.uy/~mauttone se encuentra un resumen del contenido del presente informe (cuya versión electrónica está disponible también en el sitio).

Servidor de cálculo

Es una herramienta que permite a un usuario remoto conectarse al sitio del proyecto, enviar su propio caso de prueba y obtener los resultados utilizando las implementaciones de los algoritmos realizadas en nuestro trabajo. Las principales características de la misma son las siguientes:

1 - Acceso remoto a la ejecución de los algoritmos (como alternativa a la distribución del código fuente o el programa ejecutable).
2 - Los cálculos se realizan en el servidor.
3 - El usuario no necesita permanecer conectado al sitio durante el período en que se efectúa el cálculo.
4 - Soporte de concurrencia: Pueden existir varios procesos de cálculo activos en el servidor en un instante dado.

Otras características consideradas inicialmente fueron las siguientes:

- Editor gráfico (de interfase similar a la herramienta HEIDI) que permitiera dibujar la red, asignar valores de confiabilidad a los componentes de la misma y marcar nodos terminales. Esta opción fue descartada luego de evaluar el tiempo de desarrollo que insumiría.

Implementación del sitio web

La parte del sitio que contiene el resumen del informe está construido en base a varias páginas web (una por capítulo). Las mismas están relacionadas entre sí mediante hipervínculos en los puntos adecuados. Para la construcción del mismo se utilizó la herramienta Netscape Composer (editor gráfico de páginas web que permite la generación automática de archivos en lenguaje HTML).

Implementación del servidor de cálculo

A continuación se describen las herramientas y las principales ideas utilizadas para la construcción del servidor de cálculo. Básicamente las herramientas utilizadas fueron: formularios HTML, programación CGI, lenguaje Perl, programación de shell Unix. Para lograr las características mencionadas anteriormente, las principales ideas y procedimientos utilizados son los siguientes:

1 - Acceso remoto a las implementaciones de los algoritmos (como alternativa a la distribución del código fuente o el programa ejecutable): Una alternativa manejada inicialmente fue la distribución del código de los algoritmos. Entre las principales desventajas de este enfoque se encuentra la necesidad por parte del usuario de compilar el código y por nuestra parte la necesidad de la elaboración de un completo manual de usuario para el programa. Se concluyó que la opción más conveniente (principalmente por parte del usuario) consistía en un servidor de cálculo donde el usuario pueda enviar su caso de prueba y obtener los resultados.

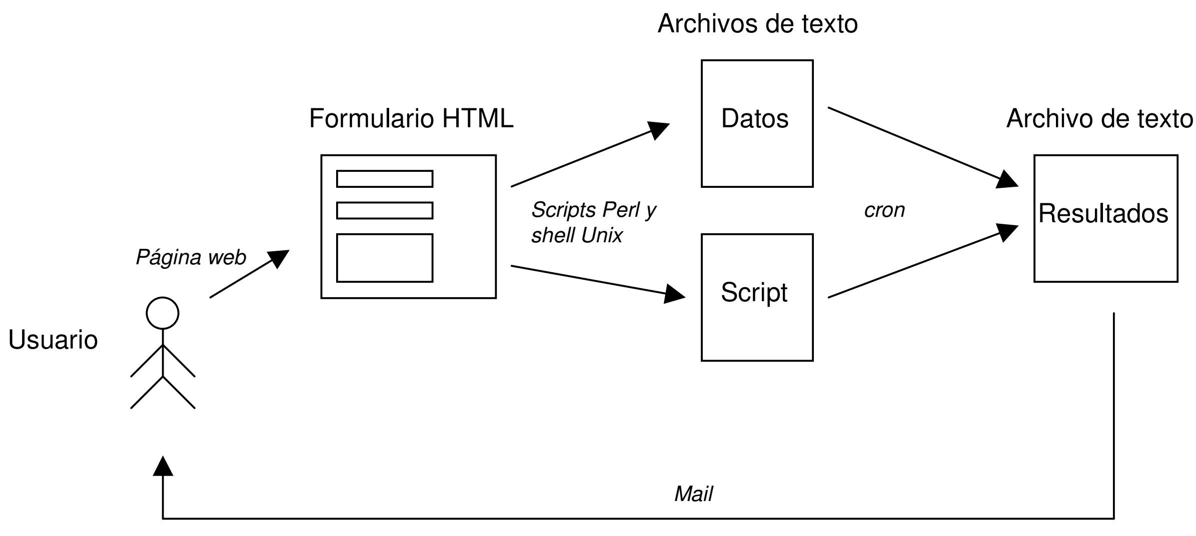
Básicamente el servidor funciona de la siguiente forma:

- El mismo corre en una red Unix, con servidor web corriendo también en sistema Unix.
- El usuario se conecta al sitio e ingresa sus datos en un formulario HTML. Los datos fundamentales son su dirección de e-mail y el texto del caso de prueba (en formato de importación/exportación de HEIDI). También debe seleccionar los algoritmos que desea utilizar para el cálculo (CGE, MCC y RVR; para los dos últimos debe especificar la cantidad de replicaciones).
- Al confirmar la solicitud del cálculo se invoca un script escrito en lenguaje Perl que realiza las siguientes acciones: crea un archivo con los datos del usuario que solicita el cálculo, crea otro archivo con los datos del caso de prueba para el cálculo y genera un archivo con código shell-script. Dicho script realizará las siguientes acciones: enviar un e-mail al usuario que solicitó el cálculo confirmando su solicitud, realizar los cálculos solicitados y enviar un segundo e-mail con los resultados obtenidos.
- Los scripts generados durante la solicitud del cálculo se colocan en un directorio determinado y cada 30 minutos se ejecuta un proceso a través del comando *cron*, que revisa si hay nuevas solicitudes y ejecuta dichos scripts.

2 - Los cálculos se realizan en el servidor: Un enfoque completamente distinto al escogido (y considerado inicialmente) fue traducir todo el código de los algoritmos, originalmente escrito en C++, al lenguaje Java y construir un applet que realizara los cálculos. La principal desventaja de este enfoque es que los programas escritos en Java se ejecutan en el cliente, y considerando que algunos algoritmos con casos de prueba de tamaño considerable insumen un gran tiempo de cálculo, esto obligaría al usuario a permanecer conectado al sitio durante el período de tiempo en que se realiza el cálculo. Por este motivo se escogió la alternativa de la programación CGI ([14] y [15]) que permite la ejecución de procesos en el servidor, lanzados por un cliente.

3 - El usuario no necesita permanecer conectado al sitio durante el período en que se efectúa el cálculo: Por lo señalado en los dos puntos anteriores, el usuario debe conectarse una vez al sitio, enviar su solicitud de cálculo, y recibirá los resultados vía e-mail una vez finalizado el mismo.

4 - Soporte de concurrencia: Los cálculos concurrentes solicitados por distintos usuarios son ejecutados por distintos scripts, por lo que la concurrencia a este nivel es resuelta por el sistema operativo. Originalmente el nombre de un script de cálculo se generaba a partir de la dirección de e-mail del usuario solicitante; esto no permitía la existencia de dos solicitudes de cálculo distintas realizadas por un mismo usuario, pendientes. Este problema se solucionó identificando una solicitud de cálculo a partir de la dirección de e-mail del solicitante y la hora (incluidos los segundos) en que fue solicitado.



Ingreso de datos, procesamiento, y envío de resultados en servidor de cálculo

8 - Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones generales del proyecto y los posibles trabajos futuros a realizarse.

Conclusiones generales

En este proyecto se ha estudiado la técnica de Reducción Recursiva de la Varianza aplicada al cálculo de confiabilidad en redes. Se ha implementado el modelo para la representación de grafos estocásticos en lenguaje C++, y tres algoritmos: el algoritmo exacto de **Generación Completa de Estados** y los algoritmos estimativos **Monte Carlo Crudo** y **Reducción Recursiva de la Varianza**. Se han generado casos de prueba para la validación y comparación de la eficiencia de los algoritmos. También se ha integrado el trabajo realizado a la herramienta HEIDI y se ha construído un sitio web para la difusión de los resultados.

Algoritmos

Las principales conclusiones con respecto a los algoritmos extraídas de las pruebas son las siguientes:

- El algoritmo **GCE** admite una implementación sencilla y da resultados exactos, pero resulta prohibitivo en términos de tiempo de ejecución para redes de tamaño medio-grande (su orden de complejidad es exponencial en la cantidad de componentes de la red). Para redes pequeñas (confiables o no) es sin dudas la opción más adecuada.
- El algoritmo **MCC** también admite una implementación sencilla y da resultados razonablemente precisos para redes de confiabilidad medio-baja. El orden de complejidad del algoritmo es lineal con respecto a la cantidad de componentes de la red. Para redes muy confiables (cualquiera sea su tamaño) deben realizarse gran cantidad de replicaciones para obtener resultados aceptables, por lo que el tiempo de ejecución aumenta en función de la confiabilidad de la red. Por lo tanto, este algoritmo es adecuado para redes de cualquier tamaño y confiabilidad medio-baja.
- El algoritmo **RVR** no es tan sencillo de implementar (se deben tener en cuenta algunas sutilezas e implementar varias funciones auxiliares). Los resultados son más precisos que para **MCC** (básicamente la varianza es menor) pero los tiempos de ejecución son mayores (el orden de complejidad es cuadrático en la cantidad de componentes de la red). Este algoritmo es adecuado para el cálculo de confiabilidad en redes muy confiables de cualquier tamaño (si bien para redes grandes los tiempos de ejecución son bastante mayores que para **MCC**, no son imposibles como en **GCE**).

Herramientas utilizadas

Para la generación de los casos de prueba se utilizó el editor gráfico de la herramienta HEIDI, donde se pudo apreciar la ventaja de contar con este tipo de herramientas.

Las pruebas se realizaron en entorno Unix, utilizando scripts que realizaban los cálculos para las distintas topologías y conjuntos de valores de confiabilidades en componentes. Los resultados se almacenaron en archivos de texto con determinado formato, que luego se procesaron en la herramienta Microsoft Excel. En relación a dicho procesamiento de datos se advirtió la necesidad de contar con una herramienta automatizada (que realizara cálculos de errores, confección de gráficos y otros), ya que se debieron realizar varias veces las mismas pruebas con un relativamente grande volumen de datos, lo que involucraba gran cantidad de tareas repetitivas que insumieron una importante cantidad de tiempo.

También se ha integrado el trabajo realizado a la herramienta HEIDI y se ha construido un sitio web para la difusión de los resultados obtenidos. Con respecto a la integración con HEIDI, se aprecia el hecho de contar con un diseño de módulos adecuado para realizar extensiones, ya que se requirió de relativamente poco trabajo para lograr la integración de los nuevos algoritmos a la herramienta. En relación a la construcción del sitio web, se apreciaron las facilidades del lenguaje Perl para la manipulación de textos; el mismo se utilizó en la construcción del servidor de cálculo, tanto para procesar los datos de entrada como para la generación de los scripts que realizan los cálculos y envían los resultados.

Trabajos futuros

Los trabajos futuros interesantes, básicamente se distribuyen en los siguientes grupos:

Implementaciones de los algoritmos

El algoritmo **GCE** admite dos mejoras importantes:

- Llevar el cálculo de la probabilidad del subgrafo (o estado de la red) acumulado y modificarlo en cada invocación (ya que lo único que varía es la presencia o no de un componente). Esto evitará la invocación a la función que calcula la probabilidad del subgrafo en cada estado generado de la red.
- Se pueden agregar predicados de poda al algoritmo de Backtracking utilizado, de forma de no examinar ciertos estados para los cuales se puede predecir que no son operativos.

Para el algoritmo **RVR**:

- Si en la ejecución de varias replicaciones del algoritmo se han recorrido todos los caminos desde la raíz del árbol de la computación hasta alguna hoja (ver ejemplo del capítulo 3) no es necesario seguir ejecutando más replicaciones. Se puede entonces incluir un control que controle esta condición de forma de no realizar replicaciones innecesarias.
- Para el caso particular de la medida $\boldsymbol{R_{st}}$ y para algunas topologías particulares se podría mejorar la performance del algoritmo condicionando el cálculo a un camino s-t y no a un corte. Esto se puede ver observando que la varianza de la variable aleatoria utilizada para estimar la medida $\boldsymbol{Q}$ es igual a $(\boldsymbol{Q}(\boldsymbol{G}) - \boldsymbol{Q_D})\boldsymbol{R}(\boldsymbol{G})$, por lo que la reducción de la varianza está directamente ligada a la probabilidad de falla del corte extendido D. De forma análoga, en el cálculo de $\boldsymbol{R_{st}}$ condicionando a un camino s-t, la reducción de varianza está ligada a la probabilidad de funcionamiento de dicho camino. Por lo observado anteriormente, para el caso particular del cálculo de la medida $\boldsymbol{R_{st}}$, se puede evaluar el uso de un condicionamiento por cortes o por caminos, dependiendo de las probabilidades del corte "menos confiable" y del camino "más confiable".

Evaluación de la performance

En este trabajo se comparó la performance del algoritmo **RVR** frente a los algoritmos **GCE** y **MCC**, y algunos resultados se compararon con los obtenidos mediante otros algoritmos en otros trabajos. En una etapa posterior sería útil realizar una comparación más completa frente a otros algoritmos, por ejemplo utilizando los datos encontrados en [19].

HEIDI, automatización de pruebas y sitio web

Las mejoras que admite la herramienta HEIDI tienen que ver básicamente con su interfase gráfica:

- Actualización de los componentes utilizados (menúes, botones y cajas de texto) a los del nuevo entorno gráfico.

- Utilización del botón derecho del mouse en el editor, para realizar algunas acciones tales como cambiar las propiedades de nodos y aristas.
- Agrupación de componentes en una red mediante ventanas.

Con respecto al procesamiento de los datos de las pruebas, una opción interesante es la construcción de macros para la herramienta Microsoft Excel con las acciones más frecuentes.

La principal mejora a incorporar al sitio web tiene que ver con el servidor de cálculo, al cual podría integrarse un editor gráfico (similar al de HEIDI) que permita dibujar la red, asignar valores a los componentes y marcar nodos terminales. Este editor puede construirse como un applet en lenguaje Java, el que sustituiría a la actual caja de texto donde debe ingresarse el texto del caso en formato de exportación/importación de HEIDI (que de no disponer el usuario de tal herramienta, debe generarlo de forma manual).

9 - Referencias

[1] H.Cancela, M. Urquhart, G. Rubino (1993) *HEIDI – Una Herramienta de apoyo para el diseño de Redes de Comunicaciones*, InCo - Facultad de Ingeniería, URUGUAY, IRISA - INRIA, Rennes, FRANCIA

[2] M. Ball, C. Colbourn, J. Provan (1992) *Network Reliability* (Reporte Técnico 92-74), University of Maryland, University of Waterloo, University of North Carolina

[3] H. Cancela (1999) *Adapting RVR simulation techniques for general network reliability models* (Reporte Técnico INCO 99.05), InCo - Facultad de Ingeniería, URUGUAY

[4] G. Friss de Kereki, M.Maneyro, F. Robledo, A. Sabiguero (1996) *Modelos de Confiabilidad en Redes*, Taller V, InCo - Facultad de Ingeniería, URUGUAY

[5] Rational Software Corporation (1997) *UML Notation Guide version 1.1*, Software Development magazine CD

[6] F. Berruti, P. Pereyra, R. Cardozo (1997) *Modelos de Confiabilidad*, Taller V, InCo - Facultad de Ingeniería, URUGUAY

[7] H. Cancela, M. El Khadiri (1996) *Series - Parallel Reductions in Monte Carlo Network Reliability Evaluation* (Reporte Técnico INCO 96.01), FRANCIA

[8] H. Cancela, M. El Khadiri (1995) *Recursive Path Conditioning Monte Carlo Simulation of Communication Network Reliability* (Publicación Interna Nº 915), FRANCIA

[9] G. Maquiel, P. Barrios (1996) *Interface para el modelado y diseño de redes de comunicaciones*, Taller V, InCo - Facultad de Ingeniería, URUGUAY

[10] H. Cancela, M. Urquhart (1995) *Elección de un método Monte Carlo para el cálculo de la medida R_v de confiabilidad en redes de comunicaciones y su implementación en la herramienta HEIDI* (Reporte Técnico INCO 95.05), InCo - Facultad de Ingeniería

[11] M. Easton, C. Wong (1980) *Sequential Destruction Method for Monte Carlo Evaluation of System Reliability*, IEEE Transactions on Reliability, Vol R-29, Nº 1

[12] A. Tanenbaum (1997) *Redes de Computadoras*, Tercera Edición - Prentice Hall Hispanoamericana

[13] A. Satyarayana, M. Chang (1983) *Network reliability and the factoring theorem*, Networks, 13:107-120

[14] Shishir Gundavaram (1996) *CGI Programming on the WWW*, O'Reilly & Associates, Primera Edición

[15] David Medinets (1997) *Perl 5 a través de ejemplos*, Prentice-Hall Hispanoamericana

[16] F. J. Ceballos (1991) *Curso de programación C++. Programación Orientada a Objetos*, Addison-Wesley Iberoamericana

[17] Patrick Valduriez (1994) *Algunas Ideas para Mejorar la Escritura de Informes Técnicos*, INRIA, FRANCIA (traducción al español por Richard Lezcano)

[18] B. W. Kernighan, R. Pike (1987) *El entorno de programación UNIX*, Prentice-Hall Hispanoamericana

[19] H. Cancela, M. Urquhart (1994) *Métodos Monte Carlo en la estimación de la confiabilidad en redes de comunicaciones -Estado del arte-* (Reporte Técnico INCO 94.06), InCo - Facultad de Ingeniería

[20] H. Cancela, M. Urquhart, G. Rubino (1996) *Network reliability evaluation by the Ahmad method* (Reporte Técnico INCO 96.03), InCo - Facultad de Ingeniería

[21] L. Petingi, M. Urquhart (1996) *Algorithms for the computation of communication network vulnerability indexes* (Reporte Técnico InCo 96.04), InCo - Facultad de Ingeniería

Apéndice 1

Observación 1

Tomando el estimador $R_{est} = \frac{1}{N}\sum_{i=1}^{N} 1_{\{G'_i\ operativo\}}$ para la medida de confiabilidad, este queda expresado en términos de *N* variables aleatorias de Bernoulli, independientes e idénticamente distribuídas, cada una con esperanza *R* y varianza *R*(1-*R*). Si *N* es suficientemente grande, por el teorema del límite central, R_{est} puede aproximarse por una variable aleatoria con distribución Normal de esperanza *R* y varianza *R*(1-*R*)/*N*. Esto significa que dados los estimadores R_{est} y Var(R_{est}) y un nivel de confianza ε se cumple:

$$\Pr\left\{R_{est} - \xi\sqrt{Var(R_{est})} \le R \le R_{est} + \xi\sqrt{Var(R_{est})}\right\} = 1 - \varepsilon$$

donde $\xi = G^{-1}\left(1 - \frac{\varepsilon}{2}\right)$ con $G(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}\,dt$

Por lo tanto el tamaño del intervalo de confianza (y por lo tanto, la precisión de la estimación) depende de la varianza.

Observación 2

Para un estimador *X*, se define su desviación estándar como $\sigma = \sqrt{Var(X)}$, y su coeficiente de variación o error relativo como $\frac{\sigma}{E(X)}$, que en el caso particular del estimador para la medida de anti-confiabilidad utilizado en **MCC** está dado por la siguiente expresión:

$$cv = \frac{\sqrt{\frac{Q(1-Q)}{N}}}{Q} = \frac{\sqrt{\frac{QR}{N}}}{Q} = \frac{\sqrt{R}}{\sqrt{QN}}$$

Si se fija un valor particular para el *cv* (por ejemplo 0,1), y considerando que en una red muy confiable el valor de *R* es muy cercano a 1, se tiene:

$$N \approx \frac{1}{(0.1^2)Q}$$

De la expresión anterior se pueden extraer dos conclusiones importantes:

- El número de replicaciones necesario para obtener un coeficiente de varianción dado para un sistema determinado es, a grandes rasgos, inversamente proporcional a su anti-confiabilidad. Para sistemas con baja probabilidad de falla, el valor de *N* será elevado.
- Independencia del tamaño y complejidad del sistema: Un sistema grande no requerirá más replicaciones que uno pequeño, para un coeficiente de variación dado.

Apéndice 2 - Cronograma

En este apéndice se detallan el cronograma genérico inicial propuesto al principio del proyecto, y los cronogramas inicial (confeccionado en Marzo de 1999) y su revisión (Diciembre de 1999) creados en la herramienta Microsoft Project.

Marzo-Abril-Mayo-Junio:

- Estudio de la temática de confiabilidad en redes (incluye repaso de los temas de Probabilidad y Estadística).
- Especificación de los requerimientos, análisis y diseño del software a implementarse para realizar las pruebas de los algoritmos.
- Primer informe.
- Implementación y testeo de las estructuras de datos.
- Estudio de los algoritmos a implementar.
- Implementación de algoritmos de simulación cruda y fuerza bruta.
- Implementación de algoritmos RVR, para determinar la confiabilidad en el modelo probabilístico de nodos y aristas para redes de considerable tamaño.
- Segundo informe.
- Búsqueda de casos de prueba.

Julio:

- Receso.

Agosto-Setiembre:

- Preparación de un plan de pruebas para la comparación de los algoritmos.
- Ejecución de las pruebas, validación del modelo y reporte.
- Tercer informe.
- Estudio de la herramienta HEIDI y su interfase gráfica.
- Integración del trabajo realizado, con la herramienta HEIDI.

Octubre:

- Construcción de un sitio web para la difusión de los resultados obtenidos.

Noviembre-Diciembre:

- Evaluación e informe final.

A lo anterior se agrega la planificación de reuniones con el tutor cada 15 días.

(cronograma incial en Project)

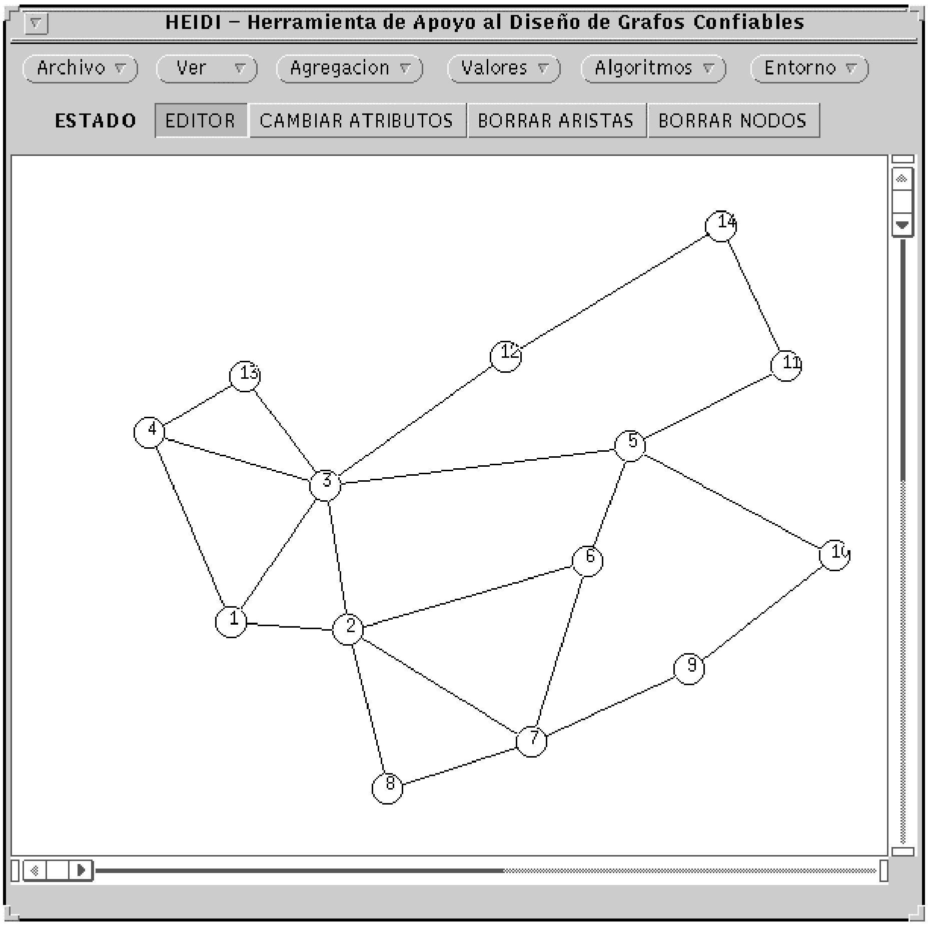
(cronograma final en Project)

Apéndice 3 - Manual de HEIDI

En este apéndice se incluye una copia del manual actualizado (incluye la integración de nuestro trabajo) de la herramienta HEIDI, generado a partir de la versión HTML del mismo.

HEIDI
Herramienta de Apoyo al Diseño de Grafos Confiables

Manual del usuario
Versión 1.0
Setiembre 1997



Proyecto BID/Conicyt 153/92
Departamento de Investigación Operativa - InCo
Facultad de Ingeniería, UDELAR, Montevideo, Uruguay

Contenido

1 Prefacio

HEIDI es una herramienta de apoyo al diseño de grafos confiables, desarrollada en el Departamento de Investigación Operativa del InCo, Facultad de Ingeniería, Universidad de la República, en el marco del proyecto BID/Conicyt 92-153 ``Herramienta de diseño de redes de comunicación confiables" (y con apoyo del PEDECIBA Informática y del Programa ECOS de cooperación científica entre Francia y Uruguay).

HEIDI provee un editor de redes y un conjunto de operaciones que permiten calcular distintas características de la red, así como buscar topologías alternativas con vistas a mejorar la confiabilidad [9,5]. HEIDI está disponible bajo el sistema operativo SunOS 4.1. Por más información sobre el equipo y el proyecto HEIDI, así como sobre los aspectos teóricos de las operaciones disponibles en la herramienta, es posible consultar la página WEB del Depto. de Investigación Operativa, situadas en la dirección *http://www.fing.edu.uy/~cancela/io/io.html*.

2 Introducción

Para utilizar HEIDI, deberá copiar el programa heidi en su directorio de trabajo o en un directorio que figure en el path del sistema. El programa se invoca ya sea desde la linea de comandos, o desde el manejador de archivos de OpenWindows, realizando un solo click con el mouse sobre el ícono del archivo ejecutable (heidi).

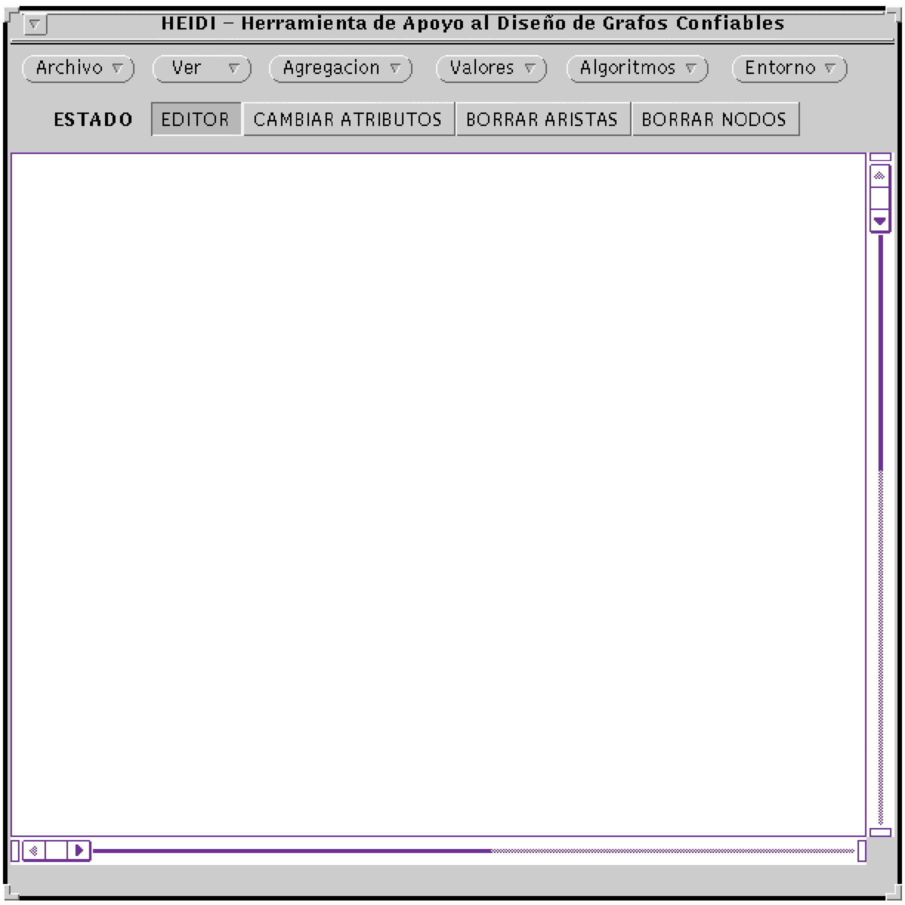


Figura 1: Ventana principal de HEIDI

La figura 1 muestra la interfaz gráfica de HEIDI, la cual está organizada en dos partes, el **área de comandos,** , la cual está situada en la parte superior de la pantalla, y el **área de edición de grafos**.

2.1 Área de comandos

Dentro de esta área, encontramos:

- Menú de opciones, se encuentra en la parte superior y realiza las operaciones de archivos, configuración del ambiente, cáclulos sobre el grafo editado. etc. En la próxima sección se describen cada uno de los menús en forma detallada.
- Control del estado de edición, el cual se encuentra en la parte inferior. Selecciona el estado de edición, el que varía según si se quiere dibujar, borrar o ponderar elementos del grafo (nodos y/o aristas).

2.2 Área de edición

En esta área se observa las barras de desplazamiento vertical y horizontal, además de la propia ára de edición, en la cual se puede dibujar, borrar, mover, modificar y ponderar los componentes de un grafo (nodos y aristas).

3 Menú

3.1 Menú de Archivo

Realiza las operaciones de grabación y recuperación de grafos a disco, además de la operación de salida.

- **Nuevo**: crea un grafo nuevo. En el caso de que se esté editando y se seleccione esta opción, se despliega un cuadro de diálogo, que pide confirmación para borrar el grafo en edición. Si se desea grabar previamente el grafo, debe elegirse la opción No del cuadro, almacenar el grafo (con la opción Recuperar/Salvar del menú de Archivos), y ejecutar nuevamente la opción Nuevo.
- **Recuperar/Salvar**: graba o recupera un grafo. Al seleccionar esta opción, se despliega en pantalla una ventana donde se pide el nombre del archivo, luego de ingresado el nombre presione ENTER y seleccione el botón correspondiente a la operación a realizar (Salvar o Recuperar).

- **Exportar/Importar**: exporta o importa un grafo. Al seleccionar esta opción, se despliega en pantalla una ventana donde se pide el nombre del archivo, luego de ingresado el nombre presione ENTER y seleccione el botón correspondiente a la operación a realizar (Importar o Exportar)

- **Directorio**: permite visualizar una lista de archivos según el filtro especificado. Al seleccionar esta opción, se despliega una nueva ventana la cual nos muestra los archivos seleccionados según el filtro. Para seleccionar un conjunto de archivos con un patrón en común, muévase al campo del Filtro, digite el patrón adecuado (p.ej.: *.grf), y presione ENTER. Para ejecutar alguna de las operaciones del directorio (**Recuperar** o **Importar**) seleccione con el mouse un archivo de la lista desplegada en pantalla y presione el botón de la operación.

- **Salir**: permite cerrar la interfaz gráfica y salir de HEIDI.

3.2 Menú de Ver

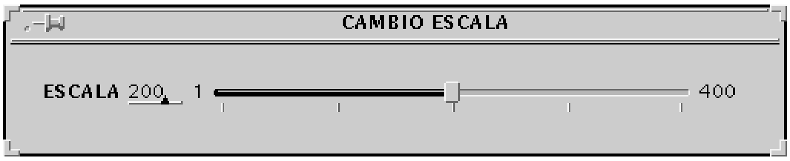
Esta opción habilita la visualización o no de los atributos de los nodos y aristas, el tipo de letra a utilizar y la escala de visualización del grafo.

- **Valores Nodos**: Configura los atributos de los nodos que se visualizarán en pantalla. Si se quiere visualizar cualquiera de los atributos de los nodos, seleccionar Valores Nodos del menú Ver y presionar el botón MOSTRAR correspondiente al atributo deseado. De forma análoga, para ocultar los valores de los atributos de los nodos, presionar el botón NO MOSTRAR correspondiente al atributo en cuestión.

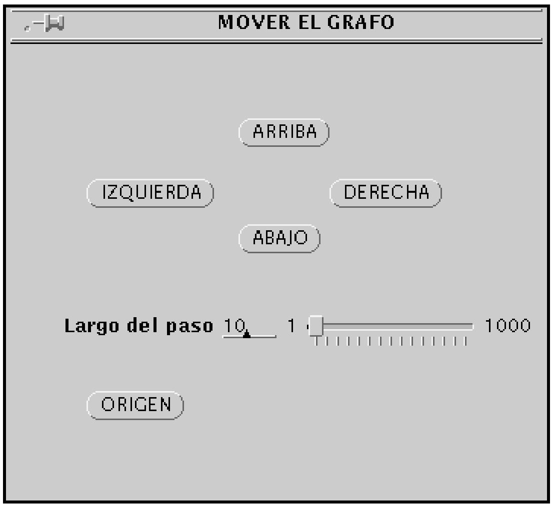
- **Valores Aristas**: Configura los atributos de las aristas que se visualizarán en pantalla. Si se quiere visualizar cualquiera de los atributos de las aristas, seleccionar Valores Aristas del menú Ver y presionar el botón MOSTRAR correspondiente al atributo deseado. De forma análoga, para ocultar los valores de los atributos de las aristas, presionar el botón NO MOSTRAR correspondiente al atributo en cuestión.

- **Tipo letra**: permite configurar la familia, el estilo y escala de la letra. Para seleccionar una nueva familia elija Tipo letra del menú Ver, se desplegará un cuadro de diálogo en el cual elegirá con el botón derecho del mouse.

- **Escala**: determina el tamaño de visualización del grafo. Seleccionando esta opción se despliega un cuadro de diálogo que permite cambiar el tamaño del grafo de dos maneras: la primera es por teclado, digitando el valor del tamaño de visualización y presionando ENTER; la segunda es arrastrando con el mouse el botón de la barra del tamaño de visualización.



- **Mover**: permite mover el grafo en pantalla, siendo posible indicar en el cuadro de diálogo el valor del desplazamiento.



3.3 Menú de Agregación

Es utilizado para crear, ver y deshacer agregaciones. Nos permite tomar un subconjunto del grafo editado y representarlo como un objeto visual único (agregación), lo cual reduce la complejidad de visualización del grafo. Una agregación es representada en la pantalla como un rectángulo de color rojo.

- **Hacer agregación**: dado un conjunto de nodos marcados realiza una agregación. Para marcar los nodos que pertenecerán a la agregación se debe seleccionar cada uno de ellos con el botón derecho del mouse. Luego de marcar los nodos elija la opción Hacer Agregación en el menú Agregación, en ese momento todos los nodos marcados se incluirán en la agregación.

- **Deshacer agregación**: deshace una agregación realizada por la opción de Hacer agregación. Para efectuar esta operación, seleccione Deshacer agregación del menú Agregación, luego posicione el mouse sobre la agregación y presione el botón izquierdo.

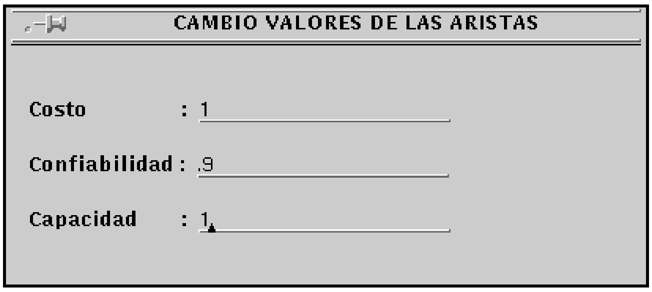
- **Ver agregación**: permite visualizar una agregación, o sea que explota el contenido de la agregación (un subgrafo). Para visualizar una agrecación seleccione la opción Ver agregación en el menú Agregación, luego posicione el mouse sobre la agregación y presione el botón izquierdo.

- **Salir agregación**: se utiliza para salir del estado de visualización (explosión) de una agregación (por lo tanto, sólo se emplea cuando está en efecto una operación de Ver agregación).

3.4 Menú de Valores

Permite atribuir valores a los atributos (confiabilidad, costo y capacidad) de los nodos y aristas en forma general. Es muy utilizado para aquellos grafos cuyos valores de atributos de nodos o aristas son iguales. Si el valor de los atributos de alguna de las aristas o nodos no coincide con la mayoría, se puede seleccionar específicamente y cambiar su valor.

- **Aristas (Todas)**: asigna valores a los atributos de todas las aristas del grafo. Para asignar valores a todas las aristas, deberá seleccionar la opción Aristas del menú Valores. Se desplegará un cuadro de diálogo en el cual se puede asignar valor al o a los atributos que se desee.



- **Nodos (Todos)**: asigna valores a los atributos de todos los nodos del grafo. Para asignar valores a todos los nodos, deberá seleccionar la opción Nodos del menú Valores. Se desplegará un cuadro de diálogo en el cual se puede asignar valor al o a los atributos que se desee.

- **Nodos (Marcados)**: asigna valores a los atributos de los nodos marcados. Para asignar valores a un conjunto de nodos, deberá seleccionar la opción Nodos (Marcados) del menú Valores. Se desplegará un cuadro de diálogo en el cual se puede asignar valor al o a los atributos que se desee.

3.5 Menú de Algoritmos

Este menú contiene las funciones de cálculo que se puede aplicar sobre el grafo. Es importante cargar los datos necesarios para las distintas funciones antes de ejecutarlas. Algunas de estas funciones modifican el grafo como por ejemplo las funciones de optimización. En estos casos es conveniente salvar el grafo original antes de ejecutar la función, y almacenar el grafo retornado con un nuevo nombre. Las demás funciones despliegan su resultado en el campo correspondiente de la **ventana de resultados** (figura 2).

Figura 2: Ventana de resultados

Nota: los campos de la ventana de resultados que aparecen en color claro reflejan resultados no actualizados y posiblemente incorrectos, ya posteriormente a su cálculo se produjo una modificación del grafo.

Las opciones del menú de Algoritmos son las siguientes:

- **Costo**: calcula la suma de los costos de todas las aristas del grafo. Es necesario que sea asignado un costo a cada arista; si una arista no tiene costo asociado, se toma por defecto el valor 0.
- **Diametro**: calcula el número de aristas que componen el camino más corto entre el par de nodos más distantes.
- **Cintura**: es el número de aristas que forman el ciclo más corto del grafo.
- **Cohesion**: es el mínimo número de aristas tales que al eliminarlas se desconecta el grafo.
- **Conectividad**: es el mínimo número de nodos tales que al eliminarlos se desconecta el grafo.
- **Conexo**: indica si el grafo es o no conexo.
- **Grado min**: es el grado del nodo de menor grado.
- **Confiab (exacta)**: es la probabilidad de que todos los nodos del grafo estén conectados (medida de confiabilidad R_V). Para calcular esta medida, cada arista debe tener asignado un valor de confiabilidad; si no se asignó valor a alguna arista, se toma por defecto el valor 0 para su confiabilidad (ver el **Menú de Valores** para asignar confiabilidades a todas las aristas). Esta opción calcula la confiabilidad de forma exacta, utilizando el método de Ahmad extendido a R_V [3].

- **Confiab (MC crudo)**: calcula la confiabilidad del grafo, utilizando simulación, y más particularmente el método Monte Carlo crudo (también llamado estándar). Al ejecutar esta opción aparece la ventana siguiente:

  que pregunta la cantidad de replicaciones que se utilzarán en el cálculo de la confiabilidad.

- **Confiab (Antit.Gr.)**: calcula la confiabilidad del grafo, utilizando el método de simulación Antitético generalizado [6,18] Al ejecutar esta opción aparece la ventana siguiente:

  que pregunta la cantidad de bloques que se utilzarán en el cálculo de la confiabilidad, y el tamaño de cada uno. El ingreso debe hacerse dando los valores requeridos separados por una barra ("/").

- **Arboles Cubr.**: el número de árboles de cubrimiento del grafo [10,13].

- **Optim (LS)**: propone un grafo alternativo al desplegado, intentando mejorar su confiabilidad (o vulnerabilidad) mediante el empleo de un algoritmo de búsqueda local simple (local search). Al ejecutar esta opción aparece la ventana siguiente:

  que permite indicar la función objetivo a optimizar en la búsqueda. Las opciones son las siguientes:

  - Función 1 (Tf0):

    ((Cintura+grMin)/(Diámetro + Xe(2)+Xn(2))) + 2*card(E)/card(X)

  - Función 2 (Tf1):

    ((Cintura+grMin+Conectividad)/(Diámetro + Xe(2)+Xn(2))) + 2*card(E)/card(X)

  - Función 3 (P0):

    ((Cintura+grMin)/(Diámetro + Xe(Cohesión)+Xn(Conectividad))) + 2*card(E)/card(X)

- Función 4 (P1):

  ((Cintura+grMin+Conectividad)/(Diámetro + Xe(Cohesión)+Xn(Conectividad))) + 2*card(E)/card(X)

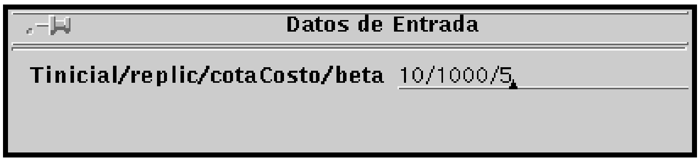
- Función 5 (H1):

  (Cintura+Cohesión*Conectividad)/(Diámetro + Xe(Cohesión)+Xn(Conectividad))

- **Optim (SA)**: propone un grafo alternativo al desplegado, intentando mejorar su confiabilidad mediante el empleo de un algoritmo de Simulated Annealing [7]. Al ejecutar esta opción aparece la ventana siguiente:

que permite indicar ciertos parámetros del algoritmo de búsqueda: Temperatura inicial, número de replicaciones, cota superior al costo del grafo, y parámetro β de descenso de temperatura. El ingreso debe hacerse dando los valores requeridos separados por una barra ("/"). En este método, la evaluación de la confiabilidad se realiza utilizando el método Monte Carlo crudo.

- **Optim (SA+Antit.)**: propone un grafo alternativo al desplegado, intentando mejorar su confiabilidad (o vulnerabilidad) mediante el empleo de un algoritmo de Simulated Annealing. Al ejecutar esta opción aparece la ventana siguiente:



que permite indicar ciertos parámetros del algoritmo de búsqueda: Temperatura inicial, número de replicaciones, cota superior al costo del grafo, y parámetro β de descenso de temperatura. La evaluación de la confiabilidad se realiza utilizando el método Monte Carlo Antitético Generalizado.

- **Confiab Rk (RVR)**: calcula la confiabilidad Rk del grafo, utilizando simulación, y más particularmente el método de Reducción Recursiva de la Varianza. Al igual que para el cálculo de la confiabilidad por MC crudo se debe especificar la cantidad de replicaciones. Para indicarse los nodos que forman el conjunto K de terminales, se deben setear con costo 1 tales nodos.

- **Confiab Rk (MCC)**: calcula la confiabilidad Rk del grafo, utilizando el método de simulación Monte Carlo crudo. También se debe especificar la cantidad de replicaciones y los nodos terminales se indican de la misma forma que para el caso anterior.

- **Confiab Rk (GCE)**: calcula la confiabilidad Rk exacta del grafo, utilizando el algoritmo de Generación Completa de Estados. Los nodos terminales se indican de la misma forma que para el caso anterior.

3.6 Menú de Entorno

- **Desplegar estado**: despliega el estado interno del editor de grafos.
- **Grafo dirigido**: configura el grafo como dirigido. En la pantalla se observa la dirección de cada arista por medio de una flecha ubicada en la mitad de la misma.
- **Grafo no dirigido**: configura el grafo como no dirigido (valor por defecto).

4 Edición

Esta sección trata sobre los estados del **control de estado de edición** y las operaciones posibles a realizarse en la pantalla de edición dependiendo de dichos estados.

4.1 Estados

La pantalla de edición posee cuatro estados diferentes, los cuales determinan las distintas operaciones a realizar sobre la misma. Estos estados son seleccionados por el **control de estado de edición**, que aparece justo sobre la pantalla de edición. Los estados son: **EDITOR, CAMBIAR ATRIBUTOS, BORRAR ARISTAS, BORRAR NODOS**. Todos los estados son excluyentes y se seleccionan posicionando el mouse sobre el estado correspondiente en el área de control de estado, y presionando el botón izquierdo.

A continuación se describen los estados con mayor detalle.

4.2 Editor

El primer estado, **EDITOR**, es el seleccionado por defecto. Permite realizar las funciones de insertar y mover nodos y aristas en el área de edición.

Para **insertar una arista** alcanza con seleccionar el estado de **EDITOR** y luego seleccionar con el botón izquierdo del mouse los dos nodos entre los cuales se desea trazar la arista. Para insertar un bucle, se presiona dos veces sobre el mismo nodo.

Para **mover un nodo** alcanza con seleccionar el estado de **EDITOR** y posicionar el mouse en el lugar donde se desea insertar el nodo, presionando allí el botón derecho. El nodo acompañará el movimiento del mouse hasta que se presione el botón izquierdo, en ese momento quedará fijo en la nueva posición.

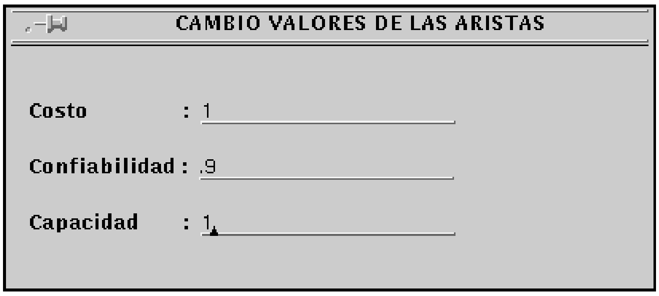
Para **mover una arista** alcanza con seleccionar el estado de **EDITOR** y luego seleccionar la arista colocando el mouse en un extremo de la misma y presionando el botón derecho; se posiciona el mouse en el nuevo nodo extremidad, y se presiona el botón izquierdo para soltar la arista.

4.3 Cambio de atributos

Tanto las aristas como los nodos del grafo poseen atributos, los cuales son: etiqueta, confiabilidad, costo y vulnerabilidad. Y cada uno puede tener un valor asignado específico para cada nodo y arista del grafo. O sea que un nodo (o arista) en particular puede tener por ejemplo un valor distinto de algunos de los atributos al de otro nodo (arista) cualquiera.

Por lo tanto existe una manera de cambiar el valor de estos atributos para un elemento del grafo específicamente. En el caso del **cambio de atributos de un nodo**, basta con seleccionar el estado **CAMBIAR ATRIBUTOS** del control del estado de edición y seleccionar el nodo posicionando el mouse sobre él y presionando el botón izquierdo. Aparecerá un cuadro de diálogo que permite elegir el atributo correspondiente y ponderarlo. Para cambiarse de atributo utilice el mouse y luego de ingresar el valor presione ENTER (en todos los atributos).

En el caso del **cambio de atributos de una arista**, basta con seleccionar el estado **CAMBIAR ATRIBUTOS** del control del estado de edición y seleccionar la arista posicionando el mouse sobre uno de sus extremos y presionando el botón izquierdo. Aparecerá un cuadro de diálogo que permite elegir el atributo correspondiente y ponderarlo. Para cambiarse de atributo utilice el mouse y luego de ingresar el valor presione ENTER (en todos los atributos).



4.4 Borrado de elementos del grafo

El editor posee dos estados que permiten borrar un elemento del grafo. Estos estados diferencian el borrado de un nodo o de una arista.

Para **borrar un nodo** se selecciona el estado **BORRAR NODO** del control del estado de edición y luego se elige el nodo a borrar posicionándose sobre él con el mouse y presionando el botón izquierdo.

Para **borrar una arista** se selecciona el estado **BORRAR ARISTA** del control del estado de edición y luego se elige la arista a borrar posicionándose sobre uno de sus extremos con el mouse y presionando el botón izquierdo.

5 Problemas conocidos

Los siguientes problemas han sido detectados en el paquete:

- El ingreso de cualquier dato debe ser completado presionando la tecla Enter antes de elegir los botones de aceptar o continuar, sin lo cual el dato no es considerado por el programa. Esto se da tanto en los ingresos de un dato único (como parámetro de un cálculo, o nombre de archivo, etc.), como en los ingresos de múltiples datos (valores de aristas y nodos).

- Diferencias de funcionamiento en la asignación de valores al costo, confiabilidad y capacidad de las aristas y nodos: cuando se asigna valor vía el menú de asignar costos a todas las aristas, todos los nodos, o todos los nodos seleccionados, esos valores son números reales; en cambio, cuando se utiliza la funcion cambiar atributo del editor, es posible ingresar no solo valores reales sino tambien cualquier texto (para realizar los calculos, los textos corresponden a valores de 0). Este comportamiento tambien se da en la grabacion y exportacion/importación de grafos, en los cuales es posible que las ponderaciones de nodos y aristas sean textos arbitrarios.

- El orden de los valores costo, confiabilidad y capacidad es distinto para las aristas (costo, confiabilidad y capacidad) y para los nodos (confiabilidad, costo y capacidad). Esto puede aparejar confusiones.

- Para importar un grafo, es necesario que cada nodo tenga una etiqueta diferente; al exportar un grafo, es posible que esta condición no se cumpla, y luego no sea posible importarlo nuevamente en la herramienta (este problema no aparece con las opciones de Salvar y Recuperar).

- El formato de exportación de un grafo no indica si el mismo es orientado o no orientado.

- Aunque es posible editar grafos orientados y no orientados, la algoritmia utilizada sólo es correcta en el caso no orientado (si se utiliza en un grafo orientado, implícitamente se elimina la orientación).

- El cálculo de confiabilidad exacta por el método de Ahmad tiene un error que ocasiona la caída del programa cuando se desea evaluar un grafo con solo dos nodos.

- El ingreso de parámetros de los métodos de cálculo no controla la existencia de todos los parámetros requeridos: si el usuario no los ingresa en el formato indicado, se toma por defecto el valor 0 para los mismos, aún cuando no es el más adecuado.

- El editor de grafos permite ingresar multigrafos (grafos con múltiples aristas entre el mismo par de nodos), y también bucles (aristas de un nodo a sí mismo). La algoritmia no soporta este formalismo, por lo que los resultados de las operaciones son indefinidos en este caso (así como los resultados de exportar un multigrafo). Las operaciones de Salvar y Recuperar en cambio, están bien definidas.

- La opción Directorio del menú de Archivo sólo permite visualizar la lista de archivos del directorio de arranque de HEIDI, y seleccionar para Recuperar o Importar un archivo de esta lista; no es posible cambiar el directorio de trabajo en forma dinámica. Para recuperar un grafo almacenado en otro directorio, es necesario ingresar su ubicación manualmente (en forma absoluta, o relativa al directorio de arranque de HEIDI).

- El manejo de errores y situaciones de excepción no es muy sofisticado, reduciéndose a mensajes en pantalla y a veces en la ventana de comandos de donde se ejecutón inicialmente HEIDI. No existe una lista completa de los errores que pueden producirse durante la ejecución del programa.
- El número máximo de nodos y aristas soportados por la algoritmia es respectivamente de Nro_Max_Nodos=200 y Nro_Max_Aristas=1000; para alterar este número, es necesario modificar estas constantes en los archivos de cabezal nodo.h y grafo.h y recompilar el programa.
- HEIDI no funciona bajo el emulador del entorno Unix X-Win32.

6 Referencias

[1]
An improvement to the total hazard method for system reliability simulation. H. Cancela and M. El Khadiri. Probability in the Engineering and Informational Sciences, 10(2):187-196, 1996.

[2]
A recursive variance-reduction algorithm for estimating communication-network reliability. H. Cancela and M. El Khadiri. IEEE Transactions on Reliability, 44(4):595-602, December 1995.

[3]
Network reliability evaluation by the Ahmad Method H. Cancela Bosi, M.E. Urquhart, G. Rubino, en Proceedings de XXIII Conferencia Latinoamericana de Informatica CLEI, a realizarse en noviembre 1997 en Valparaiso, Chile.

[4]
A simulation algorithm for source-terminal communication network reliability. H. Cancela and M. El Khadiri. en Proceedings de 29th Annual Simulation Symposium, pages 155-161, New Orleans, Louisiana, April 1996. IEEE Computer Society Press.

[5]
Evaluation and design of communication networks. H.Cancela, G.Rubino, M.E.Urquhart. En Proceedings de ICIL'95, International Conference on Industrial Logistics. Ouro Preto, Brazil, December 1995. University of Southampton, UK, and Naval Monterrey School, USA.

[6]
Métodos Monte Carlo en la estimación de la confiabilidad de redes de comunicaciones - Estado del arte. H. Cancela, M. Urquhart. En Proceedings de 24 JAIIO, Jornadas Argentinas de Informática e Investigación Operativa, Buenos Aires, Argentina, August 1995

[7]
Simulated Annealing and Communication Network Optimization. H. Cancela, M. Urquhart. PANEL'95 - XV SBC - Canela, Brasil, July 1995. In Proceedings of XXI Latin American Conference on Informatics. CLEI - SBC.

[8]
Fast Monte Carlo methods for evaluating highly dependable Markovian systems .H. Cancela, G. Rubino and B. Tuffin. In 2nd International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Salzburg, Austria, Julio 1996.

[9]
HEIDI: una herramienta de apoyo a la evaluación y diseño de redes. H. Cancela, L. Petingi, G. Rubino, and M.E. Urquhart. En VIII CLAIO, ALIO (Asociación LatinoamericanoIberoamericana de Investigación Operativa) - SOBRAPO (Sociedad Brasilera de Pesquisa Operacional). pages 581-586, Rio de Janeiro, Brazil, August 1996.

[10]
Computation of two communication network vulnerability indexes. L. Petingi and M.E. Urquhart. En VIII CLAIO, Congreso Latinoamericano de Investigación Operativa, ALIO - SOBRAPO, Rio de Janeiro, Brazil, August 1996.

[11]
Simplification techniques in Monte Carlo network reliability evaluation. H. Cancela and M. El Khadiri. In VIII CLAIO, ALIO-SOBRAPO, pages 587-592,Rio de Janeiro, Brazil, August 1996.

[12]
Construcción de modelos para la evaluación de la seguridad de funcionamiento. H. Cancela and G. Rubino.In VII CLAIO, Santiago, Chile, July 1994.Resumen extendido elegido para su publicación como artículo.

[13]
Algorithms for the computation of communication network vulnerability indexes. L. Petingi and M.E. Urquhart Technical Report INCO 96-04, PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 1996.ISSN 0797-6410.

[14]
Network reliability evaluation by the Ahmad method. H. Cancela, G. Rubino and M. Urquhart. Technical Report INCO 96-03, PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 1996.ISSN 0797-6410.

[15]
Series-parallel reductions in Monte Carlo network reliability evaluation. H. Cancela and M. El Khadiri. Technical Report INCO 96-01, PEDECIBA Informática, Facultad de Ingeniería, UDELAR, 1996.ISSN 0797-6410.

[16]
Elección de un método Montecarlo para el cálculo de la medida Rv de confiabilidad en redes de comunicaciones y su implementación en la herramienta HEIDI. H. Cancela, M. Urquhart, Technical Report INCO 95-05 ISSN 0797-6410.

[17]
Simulated Annealing and Communication Network Optimization. H. Cancela, M. Urquhart. Technical Report INCO 95-02, March 1995.ISSN 0797-6410.

[18]
Métodos Monte Carlo en la estimación de la confiabilidad de redes de comunicaciones - Estado del arte- , H.Cancela M. Urquhart, Technical Report INCO 94-06. ISSN 0797-6410.

[19]
A recursive variance-reduction algorithm for estimating communication-network reliability H. Cancela, M. El Khadiri Technical Report PI860, IRISA, Rennes, september 1994.