Departamento de Investigación Operativa
Instituto de Computación - Facultad de Ingeniería
Universidad de la República (UdelaR)
Montevideo - Uruguay

Proyecto de Grado
Ingeniería en Computación

Evaluación y Optimización de confiabilidad diámetro-acotada

Jorge Coll
Diego Costanzo
Manuel Rodríguez

- Abril de 2005 -

Tutor
Dr. Ing. Héctor Cancela
Facultad de Ingeniería - Universidad de la República (UdelaR)
Departamento de Investigación Operativa - Instituto de Computación
Montevideo - Uruguay

- Resumen -

Un concepto surgido recientemente que generaliza la idea clásica de confiabilidad en redes, es el de confiabilidad diámetro-acotada, que se define como la probabilidad de comunicación exitosa entre un cierto conjunto de nodos de una red mediante caminos operacionales de largo acotado por un parámetro *D*.

El presente trabajo abarca el estudio de una gama de aspectos relacionados al problema del diseño de redes diámetro-confiables sobre un modelo basado en redes P2P. Incluye desde una discusión de posibles lenguajes basados en XML para la representación de grafos, hasta el análisis, diseño e implementación de un algoritmo de optimización de redes según este novel concepto.
Se encara también la implementación de métodos de evaluación (Generación Completa de Estados, Monte Carlo Crudo y Monte Carlo Antitético) de algunas medidas significativas para el problema como la confiabilidad diámetro acotada, el porcentaje de nodos conectados por caminos de largo menor o igual que *D*, y la cantidad de paquetes enviados por los nodos de la red.
Incluidas en el set de herramientas desarrollado, se incluyen dos aplicaciones auxiliares de gran utilidad como ser un lector y visualizador gráfico de grafos representados en lenguaje GXL, y un traductor entre los formatos de representación de grafos HEIDI y GXL.

Además de los estudios previos respecto a la teoría sobre la que trabajamos en cada una de las etapas del proyecto, y de las herramientas desarrolladas en cada caso, se incluye también un extenso análisis de resultados referente al comportamiento de las funciones de interés ante la varianza de los parámetros de decisión del problema, al comportamiento de las soluciones retornadas por los tres métodos implementados, y un estudio comparativo de tiempos de ejecución en Java y C++ de los métodos de evaluación desarrollados.
En lo que refiere a la optimización, a partir del algoritmo ideado e implementado se detectó una tendencia en la ubicación del óptimo buscado, hecho que se analiza en la parte final del trabajo.

Palabras claves: Confiabilidad diámetro-acotada, redes, evaluación, optimización, Monte Carlo, GXL, Java.

Índice general

INFORME EJECUTIVO

APÉNDICE A: LENGUAJES DE REPRESENTACIÓN DE GRAFOS EN XML

APÉNDICE B: GRAPH EXCHANGE LANGUAGE (GXL)

APÉNDICE C: HERRAMIENTA DE TRADUCCIÓN GXL TRANSLATOR

:: Informe Ejecutivo

1. Introducción

El Departamento de Investigación Operativa del Instituto de Computación -InCo- de la Facultad de Ingeniería -Fing- de la Universidad la República -UdelaR- desarrolla desde hace cierto tiempo trabajos en algoritmia de cálculo de confiabilidad y de optimización. En dichos proyectos se han considerado diversos modelos, incluyendo el modelo probabilístico de aristas y el modelo probabilístico de nodos, entre otros. En uno de estos modelos, dado un grafo G y un conjunto de nodos terminales K, se le asigna a cada arista de este grafo una probabilidad de buen funcionamiento, asumiendo que los nodos no sufren fallas. La confiabilidad global de esta red es la probabilidad de que las aristas que funcionan induzcan un subgrafo K-conexo, es decir, que existan caminos entre cualquier par de nodos de K.
Este modelo ha sido generalizado recientemente por el Dr. Luis Petingi, investigador de la City University of New York. En la nueva propuesta, la confiabilidad global de una red en este modelo es la probabilidad de que las aristas que funcionan induzcan un subgrafo cuyo diámetro en K (distancia máxima entre pares de nodos en K) sea menor o igual que un valor *D* prefijado, lo cual también implica que el subgrafo sea K-conexo, por lo que este nuevo concepto se presenta como una generalización del concepto clásico de confiabilidad. La base de este nuevo modelo es el concepto de confiabilidad diámetro-acotada, medida que evalúa la probabilidad de que para todo par de nodos de una red exista un camino operacional de largo máximo *D* que los conecta.

Dentro de estos trabajos de cálculo de confiabilidad y optimización, se engloba un proyecto marco de cooperación con el Ing. Petingi para estudiar el modelo de diámetro-confiabilidad propuesto.
En este contexto, el presente proyecto busca extender el estudio de este planteo basado en la confiabilidad diámetro-acotada a través del desarrollo de herramientas para la evaluación y optimización basadas en éste concepto. Además se propone su empleo como modelo de las redes P2P, de forma de aplicar técnicas de optimización combinatoria que permitan alcanzar una mayor probabilidad de que los nodos de la red puedan comunicarse entre sí, manteniendo en valores razonables la carga de la misma.

Un paso previo a la definición de las técnicas a desarrollar, es el planteo del modelo de redes con el que vamos a trabajar, el cual se basa en grafos simples de aristas no dirigidas. El siguiente paso es la búsqueda de un lenguaje de representación de estos grafos que se adapte a estándares internacionales, como puede ser XML [BPSMcQ 98].
La elección de este lenguaje presenta algunos desafíos, ya que en los proyectos anteriores se venía utilizando la herramienta HEIDI [CU 93] para la representación y manejo de las redes. Dados los avances que se han producido desde la creación de la misma, pareció importante para el proyecto la actualización del lenguaje de representación de grafos intentando mantener algún grado de compatibilidad con HEIDI, de forma de lograr un nivel de interoperabilidad mínimo entre los proyectos anteriores y los desarrollos que se efectúen en este trabajo.

El lenguaje elegido para la representación de grafos fue el Graph Exchange Lenguaje o GXL [HSEW 02], el cual es un estándar de grafos basado en XML. Esta elección se hizo después de una discusión de un conjunto de lenguajes para representación de grafos, la cual se plantea resumidamente más adelante en este documento, y con total detalle en el **Apéndice B** del mismo.
Elegido el lenguaje que utilizaremos para el manejo de grafos, se encaró la creación de una herramienta de traducción de archivos HEIDI a archivos GXL y viceversa, de forma de generar el grado de compatibilidad mencionado entre las herramientas de los diferentes proyectos centrados en esta temática.

Como parte de lo que es el conjunto de herramientas desarrolladas, se creó también una herramienta de visualización de grafos representados en GXL a la cual se le incorporó la posibilidad de realizar, sobre el grafo que se esté visualizando, evaluaciones de las funciones

con que trabajamos en el proyecto, que son: la confiabilidad diámetro-acotada, el porcentaje de nodos conectados por caminos de largo menor o igual que un valor de *D* prefijado, y la cantidad de paquetes enviados por lo nodos de la red según un número de pedidos promedio que llegan a cada uno de los nodos.

Entrando en lo que es la parte de implementación de algoritmos de evaluación sobre los grafos, objetivo de nuestro proyecto, se desarrollaron diferentes métodos para el cálculo de las tres funciones de interés antes mencionadas.
Los tres métodos elegidos se basan en la generación de estados para el grafo de entrada y en la evaluación de las funciones en un subconjunto de todos los estados posibles de este grafo.
El primero de ellos es la generación completa de estados -GCE- consistente en evaluar todos los estados posibles del grafo de entrada y posteriormente promediar los resultados obtenidos para cada estado. Esta técnica es exacta, pero tiene la contrapartida de requerir un alto tiempo de ejecución para la obtención de los resultados, tiempo este que crece super-exponencialmente respecto a la cantidad de nodos del grafo de entrada.
Dados los problemas de performance de la GCE, se decidió implementar dos métodos basados en simulación estadística, como son el método de Monte Carlo Crudo -MCC- y una variante de éste denominada Monte Carlo Antitético -MCA- [EKR 92].
El primero de los métodos Monte Carlo implementa la idea clásica de esta familia de métodos consistente en elegir al azar un subconjunto de estados del grafo de entrada, evaluar las funciones de interés sobre los grafos de este subconjunto, y luego promediar los resultados obtenidos para cada uno.
Este método, a pesar de producir buenos resultados siempre que se tome un número suficientemente grande de muestras (estados del grafo de entrada), tiene la contrapartida de retornar resultados con una alta varianza y por tanto, ser poco confiables. En pos de atacar este inconveniente fue que se consideró la implementación del método MCA, que precisamente busca una reducción de la varianza obtenida en el resultado devuelto mediante la utilización de técnicas estadísticas especializadas.

En base a los métodos de evaluación implementados (en una herramienta específica para casos de grafos completos, y como una extensión de la herramienta de visualización de grafos para grafos genéricos) se hicieron un conjunto de pruebas de validación importantes, las cuales fueron acompañadas de estudios referentes al comportamiento de las diferentes funciones en varios casos.

Como última parte del proyecto y haciendo uso de las herramientas de evaluación disponibles, se encaró la definición y estudio del problema de optimización de redes centrado en el modelo de diámetro confiabilidad.
Para ello se comenzó con el planteo formal del problema a atacar y, a partir de este, a la búsqueda de un método adecuado para la resolución del mismo. El problema definido fue el de buscar los valores de *D* y *CPN* (cantidad de conexiones permitidas a cada nodo en promedio) tales que se maximizara la confiabilidad de la red sin sobrepasar una restricción conocida en la carga.
La búsqueda de la heurística concluyó con la definición de nuestro propio algoritmo de optimización, que se basa en algunas propiedades particulares del problema. Este algoritmo también se implementó e incluyó en una herramienta particular, y en base a ésta se hicieron pruebas para el estudio de su comportamiento en diferentes circunstancias, de las que se obtuvieron algunas conclusiones interesantes que se resumen en la sección 6 del presente documento.

Todas las aplicaciones implementadas, además de tener su versión de escritorio, se encuentran disponibles para ser ejecutadas usando JNPL a través de la web en la página del proyecto: http://www.fing.edu.uy/inco/grupos/invop/cda2005.

Este informe ejecutivo pretende ser una presentación global de todas las actividades realizadas a lo largo del proyecto, mostrando los aspectos esenciales de cada una y dejando para documentos específicos, que se incluyen como apéndices de éste, los detalles referentes a

cada una de estas actividades. En este sentido, siguen a esta introducción una serie de secciones en las que se presentan parte los trabajos desarrollados en cada una de las tareas realizadas como parte del proyecto.
La sección 2 presenta los objetivos generales del proyecto. La sección 3 incluye la presentación del modelo de redes con que se trabajó en todo el proyecto y en que se basan los desarrollos hechos en las secciones siguientes. En la sección 4 se introducen dos herramientas auxiliares que se desarrollaron como parte del proyecto: el visualizador de grafos en formato GXL (GXL Viewer), y traductor entre los formatos HEIDI y GXL (GXL Translator). Las partes sustanciales del estudio teórico de las funciones de interés sobre grafos, los diferentes métodos para la evaluación de las mismas, parte de cómo se llevaron estas ideas a la herramienta de evaluación (CDA Evaluator), y el análisis de resultados obtenidos mediante ésta, se describen en la sección 5. La sección 6 trata todos los aspectos involucrados en el desarrollo de la herramienta de optimización (CDA Optimizer): desde la definición de nuestro algoritmo particular, pasando por algunos aspectos de su implementación, hasta el estudio de los resultados. Por último, en la sección 7 listamos una serie de conclusiones globales al proyecto que se fueron recolectando en las diferentes etapas del mismo, junto con algunas posibilidades de extensiones y trabajos futuros en algunos de campos que se trataron.

2. Objetivos generales

En un proyecto de las características y duración del que estamos presentando, podemos diferenciar tres tipos de objetivos: aquellos de carácter global y que se mantienen en todas las etapas del mismo; los que sirven de marco precisamente para la definición de cada una de estas etapas; y los que, luego de limitadas las etapas o tareas a llevar a cabo, marcan las pautas de trabajo de las mismas.

En esta sección pretendemos hacer un listado de objetivos de los dos primeros tipos que se plantearon desde el comienzo mismo del trabajo.
Entre aquellos objetivos del proyecto que sirvieron de definición de las etapas en que encaramos el mismo podemos mencionar:

- Familiarizarse con las distintas alternativas para la representación de grafos y redes, centrados principalmente en los lenguajes basados en XML. Se plantea hacer un estudio de diferentes lenguajes para la representación de grafos para posteriormente elegir uno para ser usado en el resto de las aplicaciones a desarrollar en el proyecto. Es de importancia tener en cuenta la compatibilidad con HEIDI al momento de la selección del lenguaje, ya que es imperativo el desarrollo de alguna herramienta o mecanismo que permita la utilización de la información de redes almacenadas en formato HEIDI en las aplicaciones que se desarrollen.

- Desarrollo de una herramienta de visualización de grafos representados en el lenguaje elegido. Se estudiará también la posibilidad de hacer la misma accesible vía web, y de incorporar en ella los algoritmos de evaluación para ser aplicados a los grafos que se visualicen.

- A partir de tres métodos de evaluación a determinar para las funciones de interés sobre las redes definidas en nuestro modelo, realizar un estudio de los mismos, su diseño de alto nivel y una implementación de ellos.

- Estudiar algunos aspectos del problema de optimización de redes para luego hacer un planteo del problema en términos del equilibrio entre el tiempo de vida permitido a los paquetes en la red, y la cantidad de conexiones promedio permitidas a cada nodo de la misma. Diseñar e implementar un método de optimización para el problema planteado en base al modelo de diámetro-confiabilidad presentado.

- Validar los métodos implementados a través de pruebas para diversos problemas concretos.

- Analizar los resultados obtenidos en las diferentes etapas del proyecto, particularmente en la evaluación de funciones sobre las redes y en la optimización.

Por otra parte, los objetivos globales que se siguieron independientemente de la tarea que se estuviera encarando en cada momento, son los referentes a los aspectos de desarrollo de software. Gran parte del trabajo del proyecto está centrado en el diseño e implementación de algoritmos para la solución de los problemas de evaluación y optimización sobre redes basados en los conceptos de diámetro-confiabilidad, por lo que no deja de ser un aporte mencionar como objetivos secundarios del proyecto las buenas prácticas de ingeniería de software en el proceso de desarrollo de estas aplicaciones.
Teniendo en mente que las aplicaciones a desarrollar forman parte de un proyecto más amplio y son plausibles de una extensión futura, es importante fijarse como objetivos el diseño en capas, el encapsulamiento de funcionalidad en las clases de forma de mantener un bajo acoplamiento entre la mismas y una alta cohesión interna, y la utilización de interfaces claras y bien definidas para la comunicación de datos entre las capas y con los usuarios.

3. Marco conceptual

En esta sección detallaremos todo lo referente al modelo planteado para el proyecto no solo en cuanto a la representación de redes en un formato computable para ser utilizado en los algoritmos a implementar, sino también en la presentación formal de las funciones de interés sobre las mismas, los métodos a través de los cuales lograremos evaluar estas funciones en casos complejos, y la heurística particular creada para la resolución del problema de optimización de redes diámetro-confiables.

Cada uno de estos puntos será desarrollado en las subsecciones de esta sección, comenzando por la representación de redes, siguiendo por las funciones de interés sobre las redes, pasando por los métodos de evaluación, y finalizando con el algoritmo de optimización que implementaremos.

3.1. Modelo de redes

El campo de nuestro estudio son las redes de computadoras, en particular las redes de comunicación del tipo P2P puro. Estas redes se pueden describir como un conjunto de nodos o puertos (peers) interconectados, que tienen la finalidad global de realizar algún tipo de actividad en conjunto y en las que no existe un servidor que controle el establecimiento de conexiones y el intercambio de información.

Para este trabajo las redes consideradas tienen las siguientes características:

- Todos los nodos de la red son iguales y su probabilidad de falla es 0.
- Todas las conexiones de la red pueden estar activas o no, siendo conocida la probabilidad de que la conexión esté o no establecida.
- Todos los nodos pueden funcionar tanto como terminales o como intermediarios en un pedido de información en diferentes momentos de la operativa.

La forma intuitiva y más natural de modelar una red de comunicaciones de estas características es a través de grafos.

Un grafo $G = (V, E, \Psi)$ es una terna constituida por:

- Un conjunto $V = \{v_1, v_2, ..., v_n\}$ finito, numerable y no vacío que define los vértices del grafo.
- Un conjunto $E = \{e_1, e_2, ..., e_m\}$ que define las aristas del grafo, donde $e_i = (v_j, v_k)$ tales que v_j y v_k pertenecen a V.
- La función de incidencia $\Psi: E \rightarrow V \times V$ que asigna un par de vértices extremos a cada una de las aristas de E, $\Psi(e) = (v, v')$

Además de esta definición de grafos, usaremos a lo largo de este documento algunos conceptos relacionados que pasamos a definir:

Grafo dirigido: Un grafo es dirigido cuando existe una relación de precedencia entre los nodos extremos de una arista. En estos casos E es un conjunto de pares ordenados resultante del producto cartesiano de V.

Adyacencia: Dado un grafo $G = (V, E, \Psi)$ no dirigido, diremos que dos vértices v_j y v_k de G son adyacentes si existe una arista e_i tal que $\Psi(e_i) = (v_j, v_k)$.

Grafo completo: Dado un grafo $G = (V, E, \Psi)$ no dirigido, diremos que es completo si para todo par de vértices v_j y v_k de G se cumple que v_j es adyacente a v_k.

Para los problemas de confiabilidad en redes se define también un subconjunto *K* de *V*, de nodos especiales denominados terminales. En nuestro caso *K* = *V*.

En base a estas definiciones de grafo, podemos definir un mapeo de lo que son las redes objetivo del proyecto en grafos no dirigidos, de la siguiente forma:

- Cada nodo de la red será representado por un vértice del grafo resultante. Además cada vértice del grafo puede corresponderse con un único nodo de la red.
- Cada conexión, ya sea física o virtual, entre un par de nodos de la red, será mapeada con una arista del grafo que tendrá como vértices extremo el par de nodos en cuestión.
- La probabilidad de que una conexión entre un par de nodos no esté activa será representada por una probabilidad independiente -r_e- asignada a la arista del grafo a la que se mapeó la conexión en la red.

En el contexto de las redes P2P la cantidad *N* de nodos de nuestro grafo representará la cantidad de nodos activos en la red. A su vez este grafo será completo dado que cualquier nodo activo de la red P2P puede virtualmente conectarse con cualquier otro. Por otro lado la cantidad de conexiones que realiza en promedio cada nodo (*CCP*), se traduce en una probabilidad de utilización de las aristas: si hay *N* nodos y cada uno realiza *CCP* conexiones entonces, la probabilidad de que una arista sea utilizada es *CCP*/(*N*-1) y este valor es la confiabilidad r_e de las aristas de nuestro modelo.

Con la definición de este mapeo, comenzaremos a usar en este documento y en todos los documentos anexos a éste, los términos red y grafo de forma indistinta. Lo mismo haremos con los términos nodo y vértice, y con conexión y arista.

3.2. Funciones de interés sobre redes

Planteado el modelo de representación de redes que se utilizará, estamos en condiciones de presentar de forma más detallada las funciones de interés para su evaluación y en base a las cuales se desarrollaron los algoritmos de evaluación y optimización.

3.2.1. Confiabilidad diámetro acotada

La confiabilidad en redes, en su definición clásica, puede ser descrita desde dos puntos de vista ([Maut 00] y [CU 93]):

- Solidez o resistencia a fallos: Medida basada en algunos parámetros de las redes como 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 [PU 96]).

- Confiabilidad de los componentes: 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.

En este trabajo tomaremos el segundo enfoque, considerándose una medida de confiabilidad basada en la topología de la red y en la confiabilidad de los componentes. En el caso general se asignan probabilidades de funcionamiento a los nodos y aristas de la red, pero en nuestro modelo simplificado en que los nodos no sufren fallos, consideraremos 1 el valor de esta medida de confiabilidad para todos éstos, de forma que este factor será eliminado del análisis consecuente.

En el caso general, la confiabilidad se puede definir como una función de probabilidad con la siguiente forma:

$$\Rightarrow R_K: G \times K \subseteq V \rightarrow [0,1]$$

La evaluación de esta probabilidad $R(G)$, denominada confiabilidad K-terminal, es un problema ampliamente estudiado (por ej. en [CP 01]) y NP-complejo. Esta medida está indicando la probabilidad de que cada par de nodos terminales de K esté conectado por un camino operacional, y en la práctica indica la probabilidad de que la red esté funcionando.
A partir de esta formulación general, se definen algunos casos particulares típicos que se estudian ampliamente en la literatura del tema (por ej., en [EKMR 91]):

- $K = V \quad \Rightarrow R_V: G \rightarrow [0,1]$ (confiabilidad global)
- $K=\{s,t\} \quad \Rightarrow R_{st}: G \times V \times V \rightarrow [0,1]$ (confiabilidad fuente-terminal)

En nuestro caso consideraremos como medida de confiabilidad la confiabilidad global $R(G) = R_V$, que mediría la probabilidad de existencia de caminos operacionales entre todos los nodos de la red.

Una generalización de este concepto de confiabilidad clásico, es la confiabilidad diámetro acotada ([CP 01] y [CP 04]) que introduce restricciones en el largo (diámetro) de los caminos que conectan pares de nodos en la definición del modelo. En la práctica, esta situación se da cuando tenemos restricciones en el tiempo de vida de los paquetes en la red (TTL), por ej., en los casos en que existen demoras en cada uno de los nodos y el tiempo de comunicación entre un par de nodos debe ser menor a D veces esa demora.
En definitiva, la función de confiabilidad diámetro acotada general $R(G, D)$ estaría dada por una función del estilo:

$$\Rightarrow RDA_K: G \times (K \subseteq V) \times [1..N\text{-}1] \rightarrow [0,1]$$

Esta generalización del problema de confiabilidad K-terminal, estaría midiendo la probabilidad de que todo par de nodos terminales en K esté comunicado por un camino operacional de largo menor o igual a D. En particular, si G tiene n nodos, $R(G, n\text{-}1)$ estaría midiendo la confiabilidad K-terminal clásica $R(G)$.
En concreto, la medida de confiabilidad que nos interesará evaluar en nuestro estudio será la confiabilidad diámetro acotada para el caso $K = V$, que queda definida por la función:

$$\Rightarrow RDA_V: G \times [1..N\text{-}1] \rightarrow [0,1]$$

$$\Rightarrow RDA_V(G(V, E, \Psi), D) = R \qquad \text{con } R \text{ entre 0 y 1}$$

3.2.2. Porcentaje de pares de nodos conectados

Para nuestra definición de la función de confiabilidad diámetro acotada con que trabajaremos consideramos la medida de probabilidad de que todos los nodos, o sea que $K = V$, estén comunicados entre sí por caminos de largo a lo sumo D.
Si ahora consideramos $K = \{s, t\}$, donde s y t son nodos del grafo G, podemos definir de forma análoga una función de confiabilidad RDA_{st} que indique la probabilidad de que el par de nodos s y t esté conectado por un camino de largo menor o igual a D.

$$\Rightarrow RDA_{st}: G \times \{s, t\} \times [1..N\text{-}1] \rightarrow [0,1]$$

La función RDA_{st} expresa la probabilidad de que un nodo específico de G pueda comunicarse por un camino de largo máximo D con otro nodo del grafo. Si miramos entonces el promedio de todos los valores de RDA_{st} para todas las combinaciones posibles de nodos s y t pertenecientes al conjunto de nodos del grafo, este valor es equivalente al porcentaje de nodos conectados por caminos de estas características.

Esta medida es de interés principalmente para grafos en los que su confiabilidad diámetro acotada sea menor que 1, situación ésta que es el común de los casos para grafos medianos y grandes.
Dado que en nuestro proyecto el interés está dado en el estudio de valores globales de la red, y no en valores propios de los nodos, tomaremos la función P, que se define a continuación, para el estudio de la conectividad de las redes y para la posterior implementación en las aplicaciones, en lugar de estudiar la función RDA_{st} antes detallada.

La función de porcentaje de nodos conectados por caminos operacionales diámetro-confiables, se basa entonces en la obtención de todos los pares de nodos conectados por caminos de diámetro menor o igual a *D*, y en su división por el total de pares de nodos del grafo calculado en base a la cantidad de nodos del mismo, que expresaremos como *n*.

$$\Rightarrow P: G \text{ x } [1..N] \rightarrow [0..100]$$

$$\Rightarrow \mathrm{P}(\mathrm{G}(\mathrm{V},\mathrm{E},\psi),\mathrm{D}) = \frac{[\text{Cantidad de pares de nodos conectados por caminos de largo} \leq \mathrm{D}]}{n*(n-1)/2}$$

3.2.3. Cantidad de paquetes generados por los nodos

Otra medida importante en las redes, pero no tan estudiada como las asociadas a la confiabilidad, es la cantidad de paquetes generados por los nodos.
Operacionalmente, en una red P2P por ej., esta medida sería un reflejo de la carga promedio que está teniendo la red, por lo que a lo largo de este documento y los apéndices correspondientes nos referiremos a ella como carga.

Esta medida depende, al igual que la confiabilidad, de la topología de la red y de las confiabilidades de sus componentes, a lo que se agrega el algoritmo de distribución de los paquetes en la misma. En nuestro estudio consideraremos un mecanismo de distribución de paquetes básico: la inundación implementada como un algoritmo de búsqueda BFS.
En éste, cada nodo al que le llegue un pedido de información (externo al sistema) genera un pedido de la información solicitada que se envía a todos sus vecinos. A su vez, cada uno de éstos chequea la existencia de la información requerida en su base de datos y, en caso de no encontrarse la misma, se reenvía el paquete a todos sus adyacentes (menos a aquel del que llegó el pedido), continuando la diseminación del pedido de forma recursiva.
Se podría agregar "más inteligencia" al este algoritmo, pero la simplicidad de éste nos permite buscar fórmulas analíticas simples para la cantidad de paquetes circulando en un momento particular, de forma de contar con valores para comparar los resultados que se obtengan mediante la simulación del funcionamiento del sistema.

Para el mecanismo básico de inundación que describimos, una medida de la cantidad de paquetes generados por cada nodo, tomando como un parámetro de entrada la cantidad de pedidos de información que llegan a cada nodo de la red en promedio -*CPN*-, estaría dada por una función de la forma:

$$\Rightarrow FC_G: G \text{ x } [1..N] \text{ x } \mathbb{R} \rightarrow \mathbb{R}$$

que para el desarrollo del algoritmo de optimización aproximamos por la función analítica:

$$\Rightarrow FC_G(G(V,E,\psi),D,CPN) \cong N.CPN\,\frac{CCP(N-1)(1-CCP^D(1-(1/N-1)^D))}{(N-1)-CCP(N-2)}$$

En lo que resta de este documento consideraremos una versión simplificada de esta función -*FC*- en la no consideramos el factor *N.CPN*, pudiendo eliminar entonces el parámetro *CPN*.

3.3. Métodos de evaluación

Presentadas las funciones de interés sobre los grafos, pasaremos a describir algunas características de los métodos a través de los cuales evaluaremos las mismas. Estos algoritmos se basan en métodos de simulación estadística en los que a partir de la descripción genérica de una red, dada por un grafo con la forma descrita en la sección 3.1 en el que cada arista tiene asociada una confiabilidad r_e, se generan diferentes estados de la misma y se evalúan las funciones de interés detalladas en la sección anterior. El resultado de la evaluación de cada una de estas funciones, saldrá entonces de algún promedio entre los valores obtenidos para el conjunto de estados sorteados y evaluados para el grafo genérico de entrada.

Los métodos descritos en esta sección se distinguen básicamente en la forma en eligen el conjunto de estados del grafo genérico de entrada que se evaluarán, y en la forma en que se promediarán los resultados obtenidos en cada uno para formar el resultado final de la evaluación.

3.3.1. Generación Completa de Estados (GCE)

Un método intuitivo y sencillo de realizar la evaluación de una función particular sobre un grafo genérico de entrada, es tomar todos los estados posibles del mismo, evaluar la función de interés sobre cada uno de ellos, y retornar la suma de los resultados obtenidos ponderada por la probabilidad de ocurrencia de cada estado. Precisamente esta es la idea de la Generación Completa de Estados.

Este método tiene como principales ventajas la simplicidad de la algoritmia que trae aparejada, y el hecho de retornar el valor exacto de la función que se evalúe para el grafo en cuestión. Estas ventajas se ven seriamente contrarestadas por los altos tiempos de ejecución requeridos para la evaluación de todos los estados del grafo.
Dado un grafo $G(V, E, \Psi)$, la cantidad de instancias a evaluar para el mismo está dada por $2^{\#(E)}$, ya que cada arista del grafo en una instancia particular puede estar en dos estados: activa o inactiva. Teniendo en mente además que nuestro trabajo se centra en grafos completos, la cantidad de aristas del grafo queda definida por $\#(E) = \#(V)\ (\#(V) - 1) / 2$, lo que define un crecimiento exponencial de la cantidad de estados a evaluar.
Esta limitante en el espacio de grafos sobre el que podemos aplicar este método nos obliga a buscar formas de reducir el conjunto de instancias que se evalúan, tratando de lograr un buen equilibrio entre el tiempo de ejecución y la calidad de la solución retornada.

3.3.2. Monte Carlo Crudo (MCC)

Surge como la alternativa natural en la búsqueda de la reducción en la cantidad de estados generados para la evaluación, el método de Monte Carlo en su forma básica que denominamos Monte Carlo Crudo.

Este método consiste en la generación aleatoria de un conjunto de instancias del grafo genérico de entrada sobre las que posteriormente serán evaluadas las funciones de interés.
El factor determinante del algoritmo es la elección del número de instancias o muestras -*M*- que serán generadas y evaluadas. Un valor muy bajo de *M* redundará en bajos tiempos de respuesta, pero en resultados de alta incertidumbre (altas varianzas), mientras que la elección de un valor alto de *M* producirá mejores resultados a costa de un mayor tiempo de procesamiento.

En definitiva, el método calcula un estimador R^* del valor R de la función que se esté evaluando, el cual está dado por:

$$\Rightarrow R^* = \frac{1}{M}\sum_{i=1}^{M}\phi(G_i)$$

en donde la función evaluada está dada por ϕ, y las los G_i son realizaciones aleatorias independientes del estado del grafo que modela la red, donde cada arista del mismo está presente o no según el modelo de probabilidad descrito anteriormente.
Dado que esta fórmula es un estimador del valor real de la función, resulta de interés determinar la calidad de este estimador a través de la obtención de su varianza y, a partir de esta, los intervalos de confianza.
La varianza para el estimador que definimos -R^*- sería entonces:

$$\Rightarrow Var(R^*) = \frac{\sum_{i=1}^{M}\phi(G_i)^2}{M(M-1)} - \frac{R^{*2}}{M-1} = V$$

Las ventajas de este método pasan por la posibilidad de evaluar funciones en grafos en donde la GCE es inviable. También se pueden obtener resultados con un grado determinado de calidad (en el sentido de proximidad con el valor real *R* de la función) a partir de la correcta elección de la cantidad de muestras *M*.
La principal desventaja de éste método pasa precisamente por la alta varianza de la solución obtenida en los casos en que el número de muestras considerado no es muy alto. El último método que presentaremos busca atacar esta falencia del MCC.

3.3.3. Monte Carlo Antitético generalizado (MCA)

Conocido el problema de calidad de los resultados obtenidos mediante el método de Monte Carlo Crudo, se han estudiado diferentes técnicas en pos de una reducción de la varianza de los resultados que se obtienen, de las que podemos mencionar la Reducción de Varianza Recursiva ([Maut 00] y [CEK 94]).

El método que utilizaremos en este proyecto es el denominado Monte Carlo Antitético generalizado, desarrollado en [EKR 92] e inspirado en una generalización del método de variables antitéticas. El uso de éstas tiene beneficios tanto en tiempo de ejecución como en la precisión de las soluciones que se obtienen en comparación con el MCC.
El algoritmo sigue la idea planteada para el MCC de tomar un número predefinido de estados del grafo genérico, solo que cambian los algoritmos de generación de estos estados.

Por detalles en éste método, dirigirse a la sección E.2.3.3 del **Apéndice E**, o al mencionado paper [EKR 92].

3.4. Algoritmo de optimización

Una medida interesante en el marco del modelo basado en redes P2P planteado, es la cantidad máxima de nodos por la que puede pasar un paquete de pedido de información antes de ser descartado (o vista de otra forma, la cantidad de reenvíos que puede sufrir un paquete antes de su descarte) que se conoce como time-to-live o TTL. Aumentar el TTL implicaría aumentar el tráfico de paquetes de la red, disminuyendo entonces la eficiencia global de la misma; mientras que una disminución de este valor podría dejar como inaccesible fuentes de información que en realidad se encuentran disponibles en la red.
Otro parámetro importante en este marco es la cantidad de conexiones promedio que realiza cada nodo de la red, que actúa de forma similar que el TTL: un aumento de la misma radica en un aumento de la carga de la red, mientras que una disminución podría mostrar como no accesible información a la que se podría acceder con la posibilidad de más conexiones.

En estos sistemas hay un claro tradeoff entre la minimización de la carga de la red y la probabilidad de que un nodo pueda acceder a información que brinda otro nodo. Este problema es el que pretendemos abordar desde la perspectiva de la optimización combinatoria utilizando algún método que resulte eficaz y eficiente en esta clase de problemas. Para ello tomaremos como base los algoritmos de evaluación de las dos medidas de interés en este contexto: la cantidad de paquetes generados por nodo (como representación de la carga en la red) y la confiabilidad diámetro acotada (como medida equivalente al TTL de la red en el marco de nuestro modelo, en el que el diámetro de un camino se mide por la cantidad de aristas que pertenecen al mismo)

El planteo formal de este problema puede hacerse desde dos enfoques diferentes que se discuten en el **Apéndice E**. Concretamente, el problema que encararemos en el proyecto está definido de la siguiente forma:

- **Parámetros de entrada:**
  - $N \in \aleph^+$ → Cantidad de nodos de la red.

- $L \in \aleph^+$ → Límite de carga de la red.
- $CPN \in \aleph^+$ → Cantidad de pedidos promedio por nodo.

- **Variables de decisión:**
  - $CCP \in \aleph^+$ → Cantidad de conexiones promedio por nodo.
  - $D \in \aleph^+$ → Largo de los caminos operacionales válidos de la red.
- **Espacio de problemas:**
  - Podemos pensar nuestro espacio de problemas como una matriz *G* de *N*-2 x *N* -1 en donde la *CCP* varía en las ordenadas y *D* en las abscisas.
- **Salida:**
  - Pareja de valores (*CCP*, *D*) que maximizan la confiabilidad diámetro acotada -CDA- cumpliendo la restricción de carga *C* < *L*.

Entrando en lo que es nuestro algoritmo de optimización en sí, el mismo se basa en 4 propiedades básicas de las funciones de evaluación involucradas en el problema:

1. La confiabilidad RDA_K es creciente para valores de *CCP* crecientes y un valor fijo de *D*.
2. La confiabilidad es creciente para valores decrecientes de *D* y un valor fijo de *CCP*.
3. La carga *FC*, medida como la cantidad de paquetes generados por todos los nodos de la red, es creciente para valores de *CCP* crecientes y un valor fijo de *D*.
4. La carga es creciente para valores decrecientes de *D* y un valor fijo de *CCP*.

En base a estas definiciones inicialmente podemos dividir los puntos de nuestro espacio de problemas entre aquellos que cumplen la restricción de carga planteada y aquellos que no. Más aun, podemos definir un conjunto de puntos *F* que denominamos frontera de carga dado por:

$$F = \{(CCP_F, D_F) \in G \mid FC(G[CCP_F, D_F]) \leq L \wedge \neg\exists D_i > D_F \mid FC(G[CCP_F, D_i]) < L \wedge \neg\exists CCP_i > CCP_F \mid FC(G[CCP_i, D_F]) < L \}$$

En el cuadro 1 se puede apreciar el espacio de soluciones para el problema cuando *N*=20, *LC*=5000 y *CCP*=1. Los puntos marcados con tonos de gris son candidatos pues satisfacen la restricción de carga, pero solo los resaltados en gris oscuro forman lo que definimos como la frontera de carga *F* y son sobre los que se buscará aquel de ellos que maximice la CDA.

CCP \ D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	20	39	57	74	90	105	120	133	146	159	170	181	192	202	211	220	228	236	244
2	40	116	259	531	1.047	2.024	3.875	7.381	14.026	26.615	50.468	95.664	181.298	343.552	650.981	######	######	######	######
3	60	231	715	2.093	6.007	17.134	48.756	138.630	394.060	######	######	######	######	######	######	######	######	######	######
4	80	383	1.532	5.885	22.382	84.897	321.796	######	######	######	######	######	######	######	######	######	######	######	######
5	100	574	2.817	13.446	63.791	302.267	######	######	######	######	######	######	######	######	######	######	######	######	######
6	120	802	4.679	26.718	151.993	864.078	######	######	######	######	######	######	######	######	######	######	######	######	######
7	140	1.068	7.225	48.055	318.822	######	######	######	######	######	######	######	######	######	######	######	######	######	######
8	160	1.373	10.563	80.217	608.122	######	######	######	######	######	######	######	######	######	######	######	######	######	######
9	180	1.715	14.800	126.373	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
10	200	2.095	20.045	190.099	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
11	220	2.513	26.404	275.380	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
12	240	2.968	33.986	386.610	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
13	260	3.462	42.899	528.590	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
14	280	3.994	53.249	706.528	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
15	300	4.563	65.145	926.043	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
16	320	5.171	78.694	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
17	340	5.816	94.005	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######
18	360	6.499	111.184	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######	######

Cuadro 1: Espacio de soluciones y su frontera para N=20 y LC=5.000

Habiéndose determinado esta frontera, sabemos que el óptimo buscado se encuentra dentro del conjunto de puntos que integran la misma, ya que cualquier otro punto con valores menores

de *D* o *CCP*, tendrá un valor de confiabilidad menor que el punto de la frontera, mientras que un punto con mayor valor de *D* o *CCP* violará la restricción de carga.
De esta forma para encontrar el óptimo buscado resta recorrer este conjunto de puntos evaluando la confiabilidad diámetro acotada de cada uno, almacenando y retornando el punto donde se de el máximo.

En concreto, el algoritmo de optimización desarrollado consta de dos etapas:

- Identificación de los puntos de la frontera de carga *F*: Se realiza mediante el recorrido de las líneas de la matriz *G* que definimos como espacio de problemas, determinando para cada una el punto límite de la misma en el sentido del cumplimiento con la restricción de carga, y el chequeo de su pertenencia o no al conjunto *F*.
- Búsqueda del óptimo dentro de *F*: Se recorren los puntos de *F* y para cada uno de ellos se evalúa la confiabilidad diámetro acotada, retornándose el punto que maximice esta medida.

Para la evaluación de la confiabilidad no hubo inconveniente en utilizar los algoritmos de evaluación implementados, dado que para los parámetros que manejamos el tiempo de ejecución de los mismos es razonable. Es de considerar también que las evaluaciones en esta caso son pocas (en el peor de los casos es *N*, mientras que en la mayoría de los casos este número es mucho menor a *N*).
Sin embargo no sucede lo mismo con la evaluación de la carga, básicamente por dos razones. Primero porque es inviable, en cuanto a tiempo de ejecución, la utilización de cualquiera de los métodos de evaluación desarrollados para la mencionada función; y segundo porque la cantidad de evaluaciones necesarias sería del orden de $N*\log(N)$ en el peor de los casos y asumiendo que se utiliza búsqueda binaria para calcular el valor de *D* límite para cada fila de la matriz definida.

Esto motivó la búsqueda de una fórmula analítica para la estimación de la carga, llegándose a la definición de la fórmula presentada en la sección 3.2.3, la cual se comprobó se comporta de muy buena manera, retornando valores que caen dentro del intervalo de confianza que definimos para la evaluación utilizando MCC.

4. Herramientas auxiliares desarrolladas

Sin perder de vista que el centro del proyecto gira en torno a lo que es la evaluación de las funciones descritas y la posterior optimización de la confiabilidad diámetro-acotada, hay algunos aspectos que resultan de gran importancia para brindar un soporte adecuado a estas actividades y a sus usuarios.
En este contexto presentaremos en esta sección dos herramientas auxiliares que fueron desarrolladas: un traductor de archivos de grafo en formato HEIDI a archivos GXL y viceversa, y un visualizador de grafos representados en éste último formato.

4.1. Traductor HEIDI-GXL

Habiéndose definido el modelo de grafos que utilizaremos para modelar la realidad objetivo, el primer paso en un proyecto con un énfasis importante en lo que es el desarrollo de aplicaciones, como es el caso del presente, es la elección del modelo de datos que se utilizará para la representación de los objetos del modelo, en nuestro caso precisamente los grafos.

En el marco de desarrollo de este proyecto, bajo la tutoría del Departamento de Investigación Operativa de la Facultad de Ingeniería de la UdelaR, se ha venido utilizando desde el año 1993 un lenguaje para la representación de grafos denominado HEIDI (Herramienta Inteligente para el Diseño de Redes de Comunicación Confiables) [CU 93]. A través de proyectos similares al presente se ha desarrollado una interfaz gráfica para la herramienta en el año 1995 y posteriormente, en el año 1996, se realizó una extensión del mismo.
De todas formas con el correr de los años, HEIDI ha entrado lentamente en desuso. Herramientas nuevas y más poderosas son requeridas para mantenerse en línea con el resto de las aplicaciones y sistemas.
Dadas las ventajas que ofrece XML, como pueden ser su flexibilidad, su posibilidad de ser comprendido tanto por un lector humano como por un programa, el hecho de tratarse de un estándar de gran difusión y uso (por una discusión más amplia sobre XML ver [Rodríguez 00]), parece una alternativa más que interesante la consideración del mismo para suceder a HEIDI como lenguaje de representación de grafos en el marco de nuestro proyecto.

Diferentes sub-lenguajes de XML específicos para la representación de grafos fueron evaluados, entre los que podemos mencionar (por una discusión completa referente a estos lenguajes, ver **Apéndice A**):

- Graph Exchange Lenguaje o GXL [HSEW 02a]: Construcciones básicas (nodos, aristas y atributos) sencillas e intuitivas, con posibilidad de construcciones más complejas (hipergrafos, hiperaristas, etc.) Posee una gama importante de aplicaciones y bibliotecas auxiliares disponibles.
- GraphML [GPG 02]: Las aristas manejan el concepto de puerto para los puntos de conexión con los nodos, a lo que se agrega la dificultad para la definición de atributos de nodos y aristas.
- Extensible Graph Markup and Modeling Lenguaje o XGMML [PK 01]: Proyecto basado en GML que agrega a la información de representación básica de la estructura del grafo, datos sobre la representación gráfica del mismo. Presenta construcciones sencillas para el manejo de atributos simples y complejos a nivel tanto de nodos como de aristas.

Después de la presentación y comparación de estos lenguajes, se decidió la utilización de GXL como herramienta de modelado de grafos a lo largo de este proyecto. La decisión se basó principalmente en la sencillez de las construcciones del lenguaje y en la posibilidad de contar con bibliotecas específicas para su manejo en Java.

Habiéndose planteado desde el comienzo del trabajo el objetivo de actualizar el lenguaje de representación de grafos, surge la necesidad de mantener la compatibilidad entre las herramientas existentes que manejan HEIDI, y las que se desarrollan actualmente basadas en GXL. En vista que no es el objetivo del proyecto la integración de herramientas, nos planteamos un nivel básico de interoperabilidad entre las mismas definido por la posibilidad de utilizar grafos definidos en cualquiera de los dos lenguajes en cuestión tanto en las herramientas ya existentes como en las que se generan en este proyecto, para lo que se adecua perfectamente la herramienta de traducción desarrollada.

El traductor implementado trabaja en los dos sentidos traduciendo tanto archivos HEIDI a GXL, como a la inversa.
En el primer caso la traducción queda definida de forma sencilla, dado que los nodos y aristas de HEIDI mapean respectivamente con nodos y aristas en GXL, mientras que los atributos de ambos objetos están predefinidos por el lenguaje HEIDI lo que facilita su pasaje a atributos GXL. Un snapshot de cómo se presenta la interfaz de la aplicación para este caso se presenta a continuación:

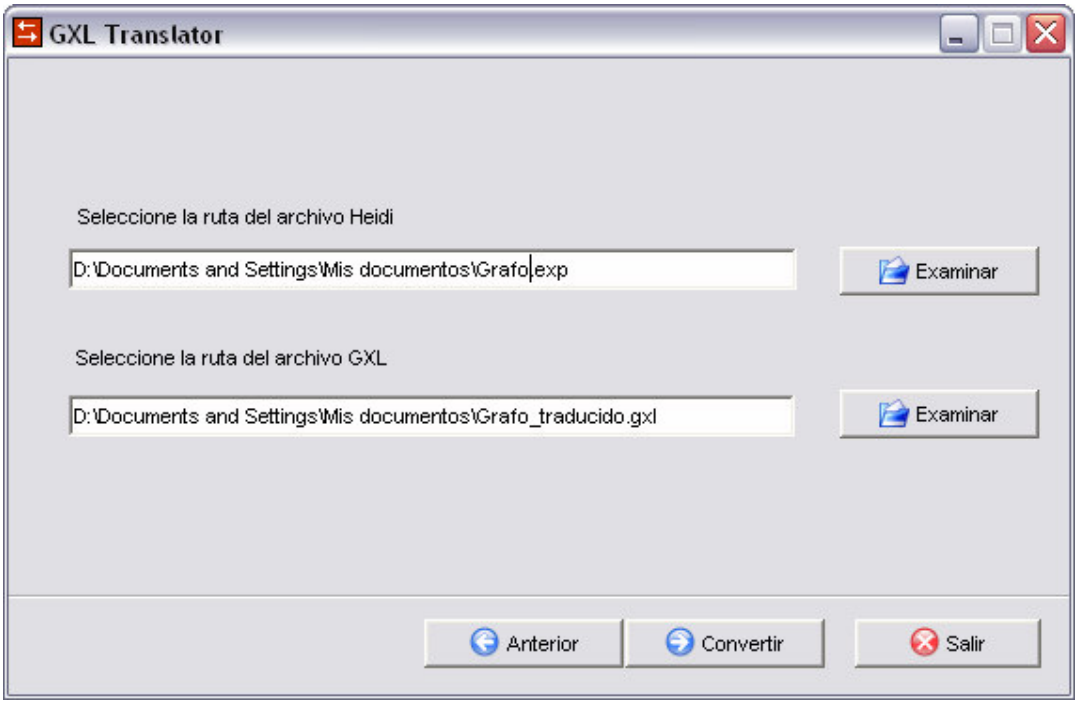


Fig. 1: Snapshot del GXL Translator: Traducción HEIDI-GXL

En el otro caso, al pasar de un archivo GXL a HEIDI, el proceso presenta dos etapas. Esta diferencia en el proceso de traducción en este sentido, radica principalmente en el hecho que los atributos en GXL son dinámicos, o sea que pueden tomar cualquier nombre y ser cualquier cantidad. La primera de estas etapas consiste en la lectura del archivo origen para la obtención del conjunto de atributos de todos los nodos del grafo y, análogamente la obtención de los atributos de todas las aristas del mismo. Partiendo de estos dos conjuntos, el usuario del sistema deberá realizar los mapeos de los atributos en cada uno de ellos a los atributos predefinidos por HEIDI, para posteriormente, en lo que sería la segunda etapa de la traducción, realizar la creación del archivo HEIDI destino con todos los nodos, aristas y atributos traducidos desde el archivo GXL.
Como hicimos anteriormente, mostramos como luce la interfaz para el caso de una traducción de GXL a HEIDI:

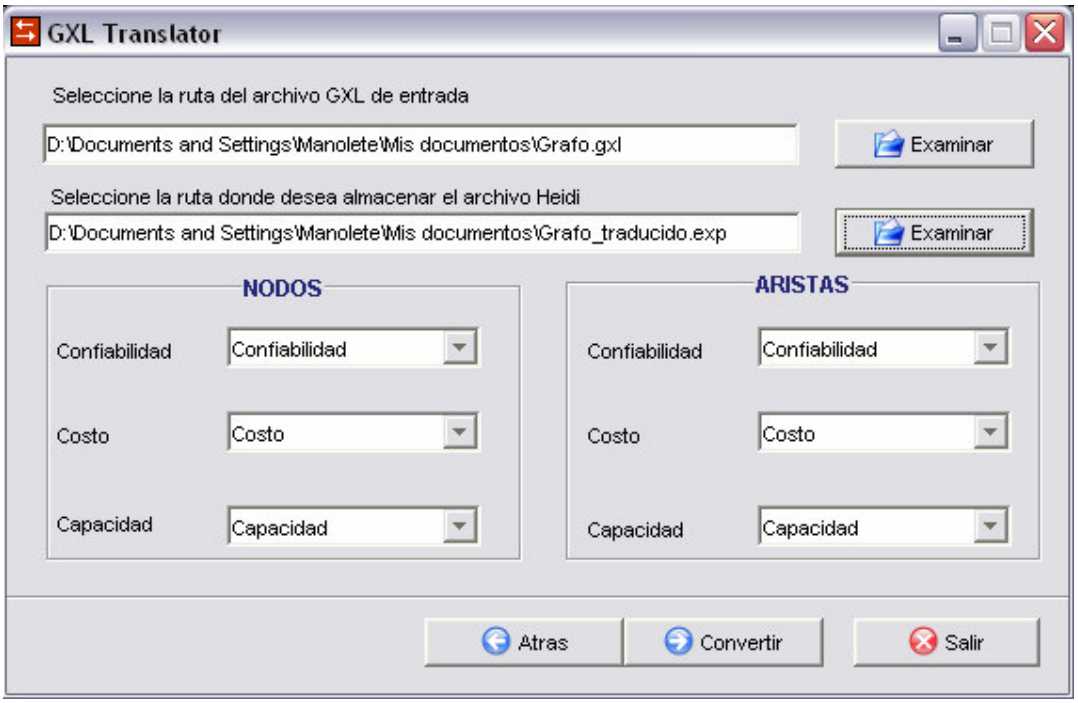


Fig. 2: Snapshot del GXL Translator: Traducción GXL-HEIDI

Esta herramienta, sencilla en su diseño, implementación y posterior uso, resulta de gran utilidad para la generación de archivos GXL para su posterior utilización en el resto de las aplicaciones que componen el proyecto.

4.2. Visualizador de grafos

Al pensar en un grafo, intuitivamente surge la idea del mismo como un objeto gráfico. De ahí la importancia, a pesar de tener definido el formato en que representaremos los mismos, de contar con algún mecanismo de visualización de la información de un grafo en formato gráfico.
Este hecho es la base que justifica el desarrollo de una aplicación específica para la visualización de grafos que encaramos como parte del proyecto.

Habiéndose elegido GXL como lenguaje para la representación de grafos, es lógico que la herramienta de visualización se base en la lectura de archivos de este formato.
Las construcciones que soporta la herramienta son las básicas definidas para grafos simples, y coinciden con las construcciones que utilizamos en GXL para los grafos con que trabajamos en el proyecto. Estas son: representación de nodos, aristas, y atributos tanto de uno como de otra.

Cabe aclarar que para la implementación de la aplicación se tomó como base los subsistemas de visualización de grafos de los sistemas *'SIM Engine'* [PIS 02] y *'EDM5'* [PIS 03] desarrollados respectivamente durante los Proyectos de Ingeniería de Software de los años 2002 y 2003. En ambos casos se utilizaron construcciones similares para la representación de estructuras de grafo tanto en memoria como de forma gráfica, las cuales se basan en la utilización de las bibliotecas externas JGraph [JGp 00] y SDA-Graph [SDA 00].
El procedimiento básico para la construcción de la aplicación fue tomar el subsistema de visualización de grafos del '*EDM5*', independizarlo del sistema y posteriormente adaptarlo a las necesidades particulares de nuestro desarrollo.
En concreto, los trabajos que se efectuaron en este aspecto abarcan:

- Independización del subsistema de visualización de grafos del '*EDM5'* de forma de transformarlo en una aplicación autónoma.

- Adaptación de la interfaz de lectura de archivos para adaptarla a la utilización de archivos en formato GXL.
- Creación de estructuras en memoria en que se representarán los grafos que se manejan.
- Rediseño de la interfaz gráfica para adecuarla a los requerimientos planteados para la aplicación.
- Agregado de operaciones específicas sobre los grafos que se manejan: visualización opcional de atributos de nodos y aristas; mantenimiento dinámico de aristas en línea recta ante movimientos de objetos en la interfaz gráfica de un grafo; almacenamiento de posiciones de los objetos del grafo en la ventana de visualización.

La aplicación en sí se basa entonces en la lectura de un archivo con la información de un grafo en formato GXL, su representación en estructuras de memoria, y su posterior presentación gráfica al usuario.

Es de gran importancia para que la herramienta cumpla su objetivo de ser una forma intuitiva y natural de acceder a la información de un grafo, que la disposición de los nodos y aristas del mismo en la ventana de la aplicación sea ordenada. En este aspecto la aplicación soporta dos modos de trabajo:

- El ordenamiento automático mediante la utilización de un algoritmo implementado específicamente a tales efectos, que se basa en una disposición de los nodos armando un árbol de cubrimiento del grafo de entrada y el posterior agregado de las aristas faltantes.
- La disposición de los objetos en la ventana en posiciones indicadas por atributos especiales en el archivo GXL de entrada.

La distinción entre los dos modos de disposición de los objetos del grafo se da de forma dinámica y transparente al usuario, siendo el factor de selección la presencia o no de los atributos especiales de posición en el GXL leído.
La presencia de estos atributos especiales, es también una construcción de la herramienta de visualización, ya que los mismos son generados por ésta al guardar un grafo que se abrió para visualizar. De esta forma se permite al usuario abrir y visualizar un grafo en un archivo GXL, arreglar la disposición de los elementos del mismo de una forma que lo satisfaga, y el posterior almacenamiento de esta disposición para una futura visualización del mismo archivo.

Continuando la línea de lo planteado para la herramienta de traducción GXL-HEIDI, presentamos un snapshot del visualizador de grafos:

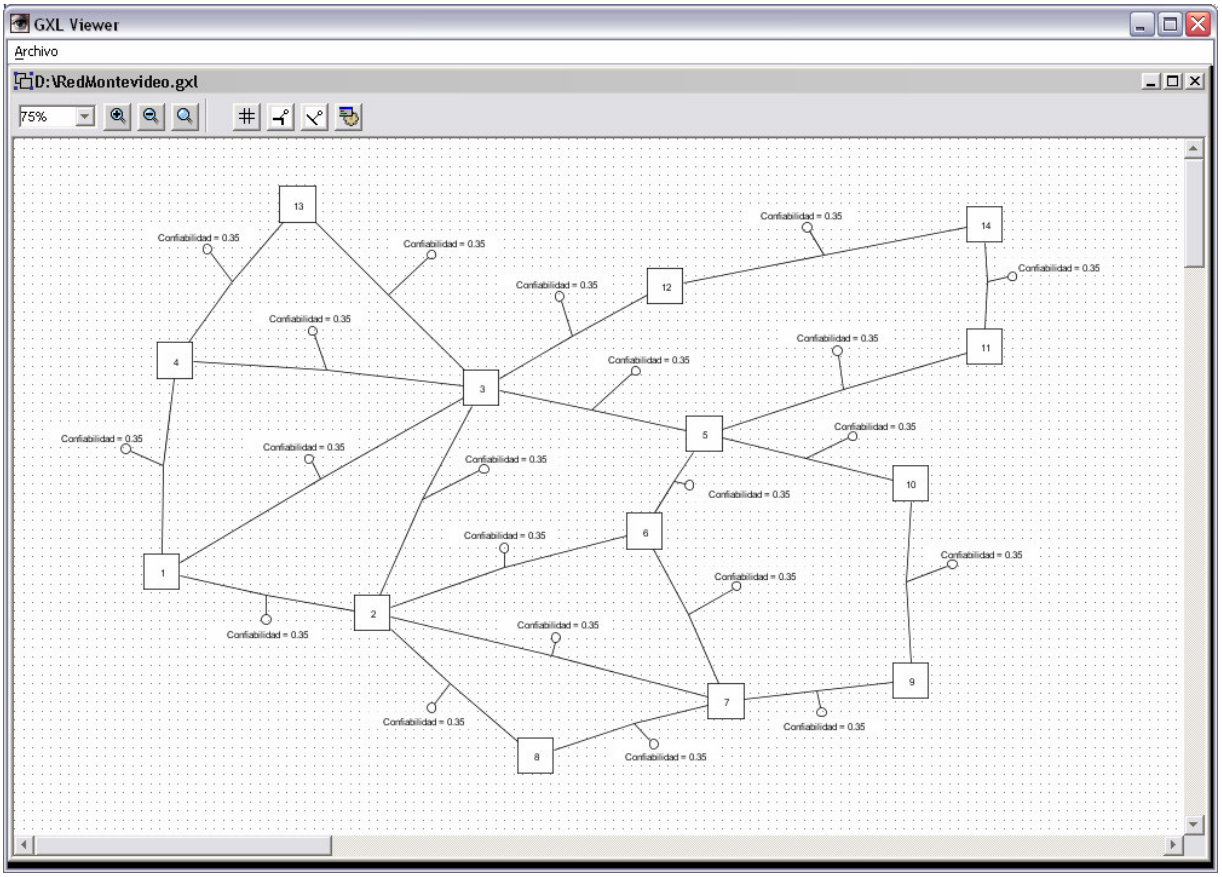


Fig. 3: Snapshot del GXL Viewer

Como se podrá observar, se añadieron a la funcionalidad básica de visualizar un grafo, algunas operaciones extra sobre el mismo como ser la selección del zoom de visualización, la activación o desactivación de la grilla de fondo, y la visualización opcional de los atributos tanto de nodos como de aristas.
Por último, en una muestra de la extensibilidad y flexibilidad de las aplicaciones implementadas, se agregó también la posibilidad de evaluación de las tres funciones de interés definidas sobre los grafos del modelo. Es de destacar que para la integración de las funcionalidades de evaluación fue necesaria la extensión de las mismas de forma de considerar grafos no completos.

Por un desarrollo detallado de lo que es el diseño e implementación del visualizador de grafos, dirigirse al **Apéndice D** de este informe.

5. Evaluación

Entrando en lo que son las aplicaciones centrales del proyecto, en esta sección haremos una breve presentación de la aplicación de evaluación que se implementó, presentación que se desarrolla en detalle en el **Apéndice E**.
En base al modelo planteado y las definiciones de las tres funciones de interés para nuestro estudio, en este punto entraremos más en algunos detalles de diseño e implementación de la aplicación.

Concretamente, el evaluador es una aplicación que, tomando como entrada un grafo válido para nuestro modelo más una serie de parámetros globales al mismo, a saber, la cota para el diámetro de los caminos válidos (*D*), la cantidad de conexiones promedio por nodo (*CCP*) y la cantidad promedio de pedidos por nodos (*CPN*), realiza la evaluación de las tres funciones definidas: la confiabilidad diámetro acotada, el porcentaje de nodos comunicados por caminos de largo menor que *D*, y la carga de paquetes enviados por los nodos de la red.

La herramienta de evaluación tiene en su desarrollo, además del objetivo de evaluar las tres funciones de interés descritas que define la creación de la misma, algunos objetivos particulares, entre los que mencionaremos:

- Calculo de valores, varianzas (en los casos que corresponda), y tiempos de ejecución para las tres funciones definidas mediante los tres métodos presentados en la sección 3.3.
- Posibilidad de selección sencilla de las funciones a evaluar y de los métodos a través de los cuales se realizará la evaluación.
- Maximización de la performance en cuanto a tiempos de ejecución y recursos del sistema utilizados.

Los tres métodos de evaluación se implementaron en Java como todas las aplicaciones del proyecto, habiéndose implementado también los dos primeros de ellos para las tres funciones de interés en C++. La idea de esta doble implementación de parte de la herramienta, es la posterior realización de un estudio comparativo de los resultados en cuanto a performance de los dos lenguajes para el problema en cuestión.

La aplicación recibe como entrada una serie de parámetros a partir de los cuales quedará definida la topología del grafo de entrada, que denominamos grafo genérico, sobre el que se evaluarán las funciones. Estos parámetros son: la cantidad de nodos (*N*), la cantidad de conexiones promedio por nodo (*CCP*), la distancia máxima que se permitirá para los caminos en el grafo (*D*) y la cantidad de pedidos por nodo (*CPN*).
A partir de estos parámetros construiremos el grafo genérico, que consiste en un grafo completo integrado por un conjunto de *N* nodos idénticos pero distinguidos por un identificador único y en que a cada arista se le asigna una probabilidad de estar activa dada por $CCP/(N-1)$.
Estos parámetros de entrada básicos, más algunos otros auxiliares a la ejecución, se muestran en la siguiente imagen que muestra la ventana de la aplicación de evaluación:

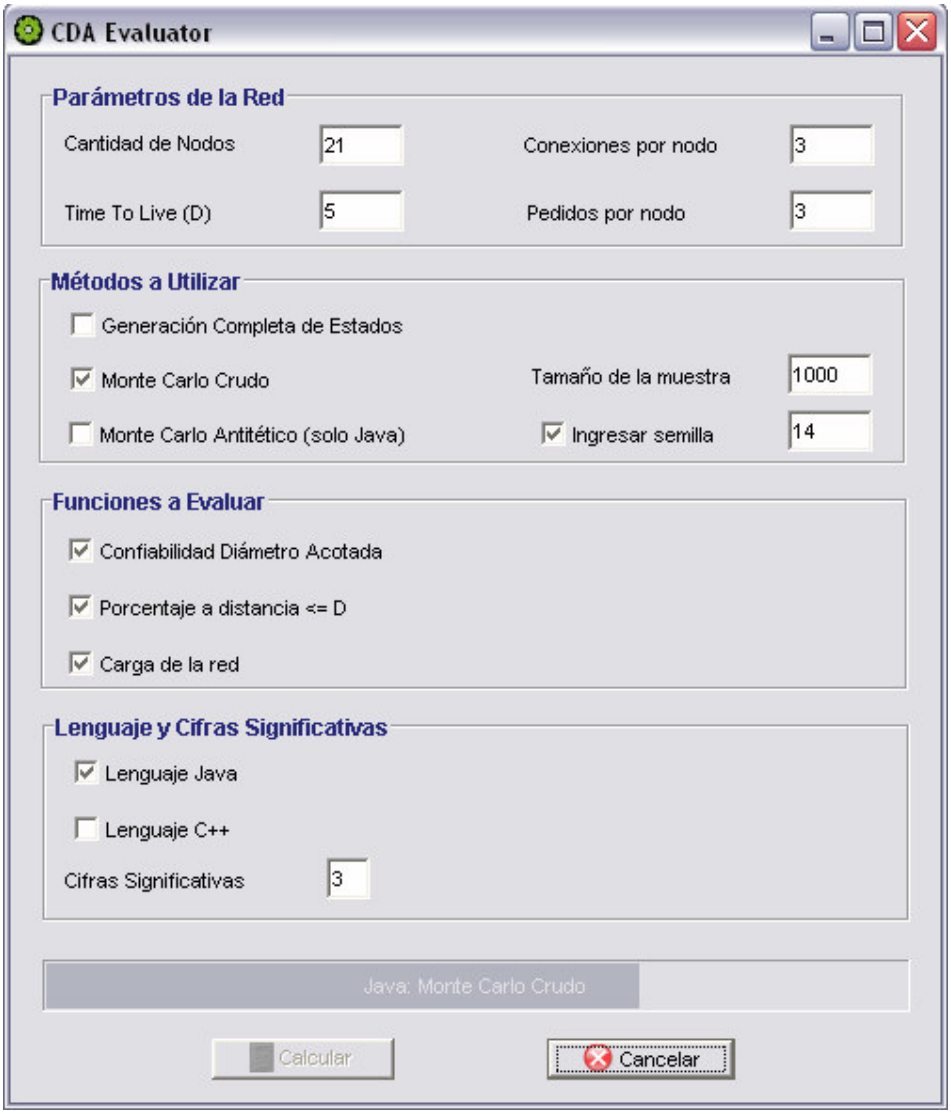


Fig. 4: Snapshot del CDA Evaluator

Definido el grafo de entrada, el proceso de evaluación en sí funciona generando estados para el mismo.
Este proceso funciona de formas diferentes según el método que se trate. Para el GCE se van generando consecutivamente todos los estados del grafo, sin tener en cuenta el valor de probabilidad las aristas. Por otro lado, para los métodos de Monte Carlo, los estados del grafo se generan sorteando un valor para cada arista y definiendo la presencia o no de la misma en el estado según el valor de probabilidad de estar activa seteado a la misma.
Para cada estado que se genere se evaluarán las funciones definidas, acumulándose estos resultados. Esta acumulación consiste simplemente en la suma de los resultados para los métodos Monte Carlo, mientras que para la GCE se suma cada resultado ponderado por la probabilidad de ocurrencia del estado que lo generó.
Luego de procesados todos los estados, que están dados por $2^{N(N-1)/2}$ para la GCE y por un valor *M* indicado por el usuario en el MCC y la MCA, se promedian los valores acumulados para los estados procesados, siendo este el valor retornado por el método.

Simultáneamente a la acumulación de los valores obtenidos por la evaluación de cada función en cada estado, se almacena también la información necesaria para el cálculo de la varianza del valor calculado para los métodos no exactos. A esto se le agrega el cálculo del tiempo de ejecución involucrado en la evaluación de cada método para cada función, el cual posee

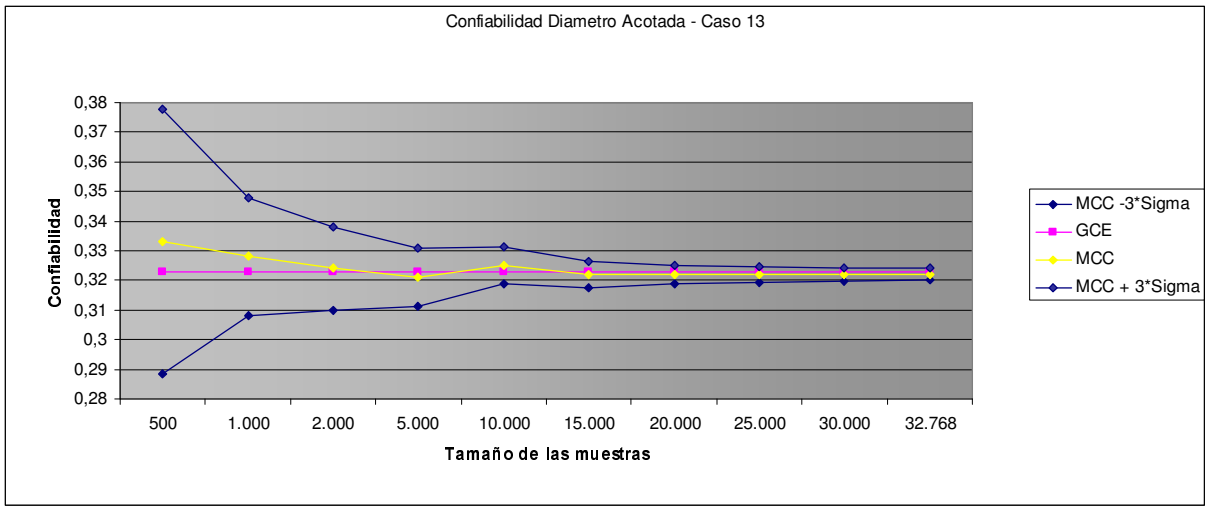
básicamente dos componente: el referente a la generación de los estados, y el de la evaluación en sí de la función sobre cada uno de los estados generados.

Un aspecto importante de toda la etapa de evaluación del proyecto, además del estudio teórico de las funciones de interés definidas, de los métodos para la evaluación de las mismas, y de la implementación de la aplicación correspondiente, es el estudio de los resultados obtenidos en pos de dos objetivos fundamentales: la validación de la implementación realizada, y el análisis en sí de los mismos.
En este sentido se hicieron una serie de pruebas, cuyo detalle ocupa toda la sección E.5 del **Apéndice E**, y que describimos esquemáticamente a continuación.

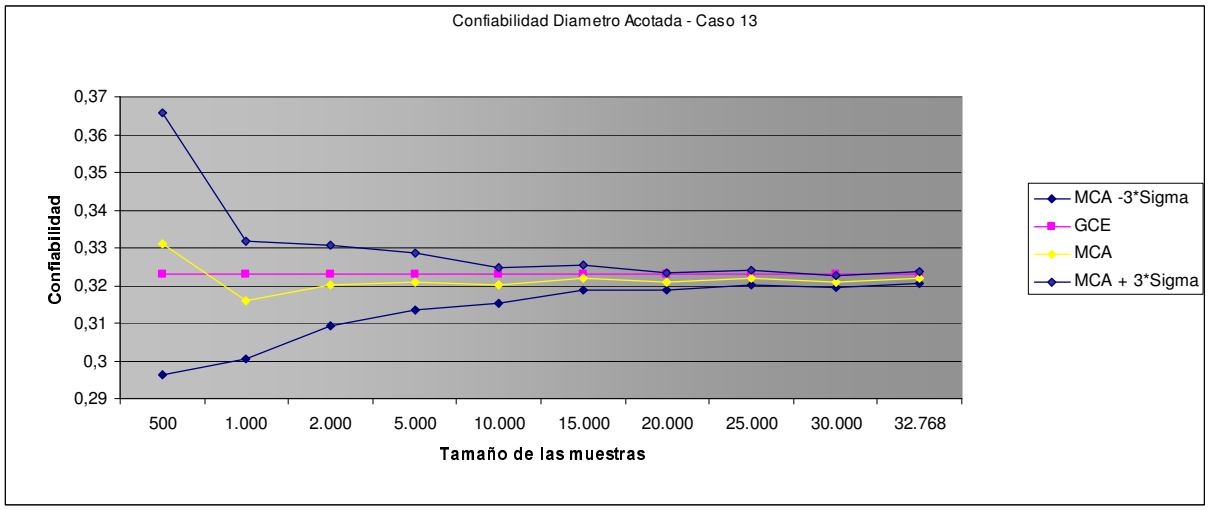
1. Validación de la GCE: Dado que este método es exacto, comenzamos haciendo una validación del mismo, básicamente mediante la comparación de los resultados obtenidos con resultados conocidos. Esta prueba se limita a la comparación con resultados de confiabilidad clásica.

2. Validación de los métodos Monte Carlo: Las pruebas en este sentido tienen dos enfoques: la comparación con resultados conocidos como en el caso de la GCE, y la comparación de los resultados obtenidos utilizando MCC y MCA contra los de la GCE. En este último caso las pruebas se restringen a casos de grafos pequeños ($N <= 7$) ya que para casos mayores las pruebas usando GCE conllevan tiempos de ejecución que las hacen inviables.

3. Comportamiento de los métodos Monte Carlo al variar el número de muestras: En la teoría, la calidad del resultado que se obtenga en un método de simulación no exacto, como son el MCC y el MCA, depende de la cantidad de estados que se evalúen. Este resultado se comprobó empíricamente para nuestra implementación, como se ve las gráficas 1 y 2, que muestran los resultados obtenidos para un caso de prueba $N = 7$, $CCP = 3$, $D = 2$ y $CPN = 1$ (caso 13):



Grafica 1: Varianza de la CDA para el MCC al variar el número de muestras



Grafica 2: Varianza de la CDA para el MCA al variar el número de muestras

Estas gráficas presentadas se reducen al estudio de un caso en particular y solo en lo referente a la función de confiabilidad diámetro acotada. Por el detalle completo de las pruebas referirse a la sección E.8.2 del **Apéndice E**.

4. Comportamiento de los métodos Monte Carlo al variar un parámetro: Entrando de lleno en el análisis de los resultados, las pruebas realizadas se centran en el estudio de las funciones variando uno de los parámetros mientras se mantienen el resto fijos. Pruebas de este tipo se repitieron variando cada uno de los parámetros de entrada, confirmándose en primera instancia los resultados que se esperan intuitivamente:

- Al variar *D* es de esperar un aumento en los resultados obtenidos en las tres funciones de interés, ya que un aumento en el largo de los caminos permitidos aumenta la probabilidad de que desde cada nodo se puedan alcanzar más nodos, incrementando entonces la confiabilidad diámetro acotada, el porcentaje de nodos conectados por caminos de largo menor o igual que *D*, y la cantidad de paquetes transmitidos por cada nodo.
- Variando *CCP* y manteniendo fijos el resto de los parámetros, el comportamiento observado es básicamente similar al comentado para la variación de *D*. En este caso, el incremento observado en los valores de las funciones se debe al aumento en la amplitud de la búsqueda realizada por cada nodo, aumento este que se torna más notorio en cada nivel de la búsqueda BFS y que se ve respaldado por los resultados observados en las pruebas.

- Por último se realizó el estudio en la variación de las funciones ante cambios en el valor de *N*. A diferencia de lo observado para *D* y *CCP*, se espera una disminución en los valores de las funciones de confiabilidad y de porcentaje.

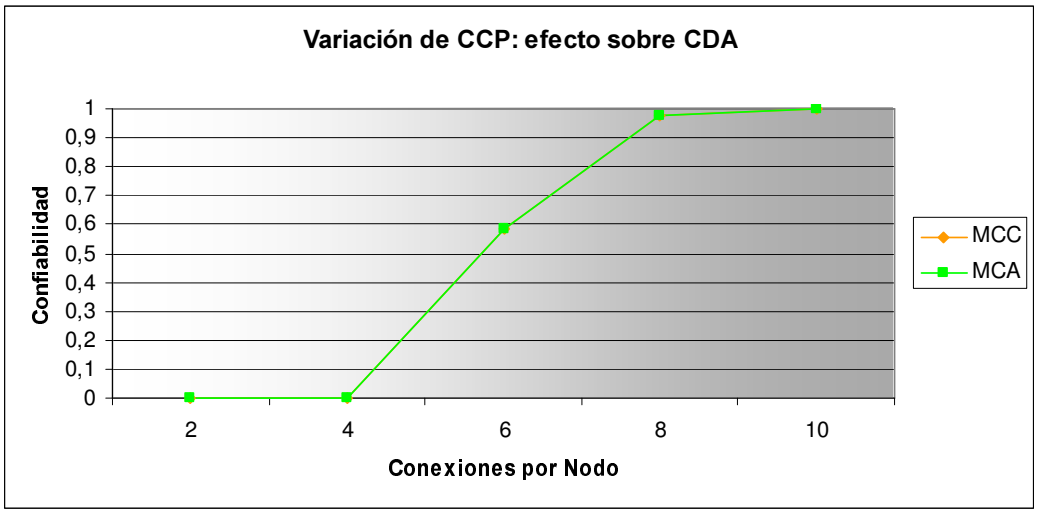
Las tres pruebas que se acaban de mencionar se realizaron para las tres funciones planteadas, efectuándose la evaluación utilizando los métodos MCC y MCA de forma de poder incluir en las mismas grafos de tamaños no despreciables.

Un hecho interesante observado es la rapidez con que tanto la CDA como el Porcentaje pasan de 0 a 1 o 100 respectivamente, para pequeños cambios en *CCP* o *D*, como se puede apreciar en los gráficos 3 y 4, los cuales se generaron respectivamente para las tablas 1 y 2 que se presentan inmediatamente antes de cada gráfica.

Parámetros fijos: $N = 50$, $D = 4$, $CPN = 1$, $M = 500$

Caso	CCP	Diámetro Acotada					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	Tiempo
37	2	0	0	0	0	0	0,016
38	4	0	0	0,141	0	0	0,187
35	6	0,584	0,022	3,11	0,584	0,0248	2,74
39	8	0,974	0,00712	5,07	0,977	0,00583	4,84
40	10	1	0	6,57	1	0	6,82

Tabla 1: Juegos de parámetros para la gráfica de variación de CDA respecto a CCP

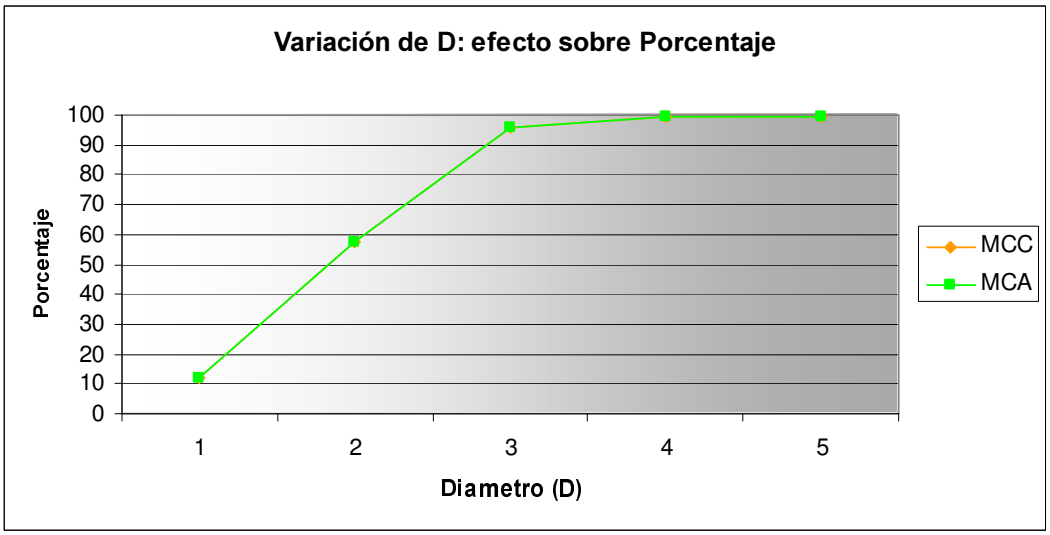


Grafica 3: Variación de la confiabilidad respecto a CCP

Parámetros fijos: *N* = 50, *CCP* = 6, *CPN* = 1, *M* = 500

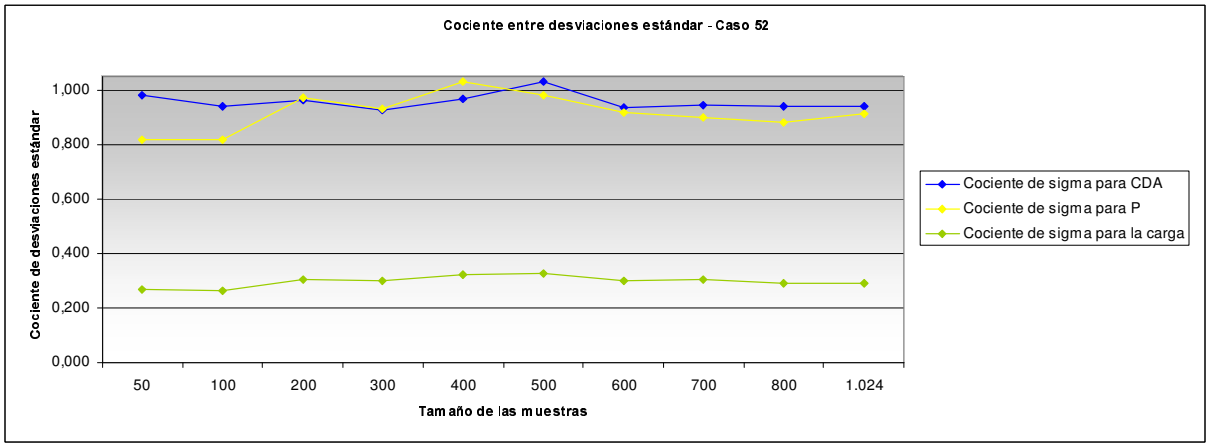
Caso	D	Porcentaje					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)
32	1	12,2	0,0417	1,2	12,2	0,00629	1,03
33	2	57,5	0,233	3,42	57,5	0,051	3,28
34	3	95,6	0,125	4,7	95,8	0,0824	4,62
35	4	99,6	0,0468	4,96	99,5	0,0551	4,82
36	5	99,6	0,052	4,93	99,6	0,0539	5,18

Tabla 2: Juegos de parámetros para la gráfica de variación del porcentaje respecto a D



Grafica 4: Variación de la función de porcentaje respecto a D

5. Comparación de los métodos Monte Carlo entre sí: El centro de estas pruebas es el estudio de la calidad de las soluciones retornadas por cada uno de los métodos Monte Carlo implementados, en el sentido de la menor varianza de uno respecto al otro. Partiendo de la base de que la principal motivación para la implementación del MCA es precisamente obtener resultados con menor varianza que el MCC, no es extraño el comportamiento observado en las pruebas, que confirman esto. Si aparece como interesante y no esperado, el mayor nivel de mejora que se observa para la función de carga respecto a las dos restantes, hecho éste que en un estudio superficial no encuentra una explicación intuitiva.
Como ejemplo de las pruebas realizadas en este punto, presentaremos uno de los casos estudiados definido por los parámetros $N = 70$, $CCP = 22$, $D = 2$, $CPN = 2$ (caso 62):



Grafica 5: Estudio de varianzas de MCC y MCA

6. Análisis de resultados en Java y C++: Para finalizar con el análisis de resultados de la evaluación se efectuó un estudio de performance de los algoritmos implementados en Java contra los implementados en C++. El enfoque planteado en esta etapa de las pruebas fue el de elegir un caso base, evaluarlo, y luego ir tomando cada uno de los parámetros y, manteniendo fijo el resto, realizar dos corridas para diferentes valores del parámetro en cuestión. De esta forma haciendo dos corridas para cada parámetro, obtuvimos un conjunto de tres valores (los de las corridas, más el del caso base) que permiten observar cómo afecta la variación de este parámetro a los resultados en cada uno de los lenguajes.

Para empezar, los resultados observados son consistentes en el sentido de que los intervalos de confianza obtenidos para ambos lenguajes se intersectan en todos los casos de prueba definidos. Entrando en el estudio de performance en sí, los resultados observados son dispares: en algunos casos los métodos en C++ son de mejor performance que los de Java, mientras que en otros casos el comportamiento es el contrario, aunque en la mayoría de los casos el comportamiento es el primero. La tendencia parece ser de que para casos de grafos pequeños a medianos, los tiempos de ejecución en Java son menores que los observados en C++, mientras que al pasar a considerar casos grandes (del orden de los 1000 nodos) los tiempos en C++ pasan a ser menores que los de Java.

Cabe la aclaración de que las conclusiones obtenidas a lo largo de estas pruebas, al ser de carácter empírico y no formal, son de un alcance limitado que nosotros de forma arbitraria generalizamos. Formalmente las conclusiones alcanzadas son de validez únicamente para los casos planteados, pero las mismas se extienden al común de los casos.

6. Optimización

Presentadas las bases teóricas del algoritmo de optimización (sección 3.4) en la presente sección entraremos en algunos detalles referentes a la implementación y al análisis de los resultados obtenidos.

La herramienta de optimización se basa en el uso de los algoritmos de evaluación implementados, específicamente en la evaluación de la confiabilidad diámetro acotada y la cantidad de paquetes enviados por los nodos (o carga).
El algoritmo presentado a alto nivel en la sección 3.4, en sí mismo es un algoritmo exacto que retorna la solución óptima para el problema planteado, pero la implementación que realizamos consiste en una heurística debido a la utilización de un método de evaluación no exacto (si bien el usuario puede elegir entre GCE, MCC o MCA, para grafos de más de 7 nodos debería seleccionar alguno de los dos métodos Monte Carlo) implementado para la CDA y una función analítica aproximada para el cálculo de la carga. Estas decisiones se justifican, en el primer caso, en la imposibilidad de utilizar la evaluación exacta, implementada mediante GCE, debido a la restricción en el tamaño de los grafos de entrada para los que la misma es computacionalmente eficiente. Por otro lado, para el caso de la evaluación de la carga, la utilización de una función analítica para su evaluación se dio para mantener la performance del algoritmo en límites razonables, ya que la simulación de la evaluación de la función de carga es la que mayores problemas de performance presenta y además es la que requiere una mayor cantidad de ejecuciones. Esto viene de la mano también, con la disponibilidad de una función analítica cuyos resultados se estudiaron en detalle y probaron ser de calidad. Específicamente, los resultados de la misma, en las pruebas realizadas, siempre estuvieron contenidos en el intervalo de confianza definido por la evaluación usando el método MCC implementado.

Los objetivos planteados al momento de encarar el diseño y desarrollo de la herramienta de optimización fueron entonces:

- Implementación del algoritmo de optimización presentado en la mencionada sección 3.4, tomando como base la utilización de los algoritmos de evaluación implementados.
- Retorno del óptimo (CCP_O, D_O,) y del conjunto de puntos (que definimos como “frontera de carga”) que son límite entre los puntos del espacio del problema que cumplen la restricción de carga y aquellos que no la cumplen.
- Desarrollo de algoritmos manteniendo al mínimo los tiempos de ejecución y la utilización de recursos del sistema, principalmente de los recursos de memoria.

Entrando en lo que es la aplicación en sí misma, ésta recibe como entrada, al igual que el Evaluador, una serie de parámetros numéricos que definen el espacio de problemas a considerar:

- N → Cantidad de nodos de la red.
- CPN → Cantidad de pedidos por nodo.
- L → Límite de carga permitido en la red, medido como la cantidad de paquetes enviados por el conjunto de nodos de la misma.
- CCP_{max} → Cota máxima para la cantidad de conexiones promedio por nodo.

Estos parámetros se distribuyen en la ventana principal del programa con el siguiente formato:

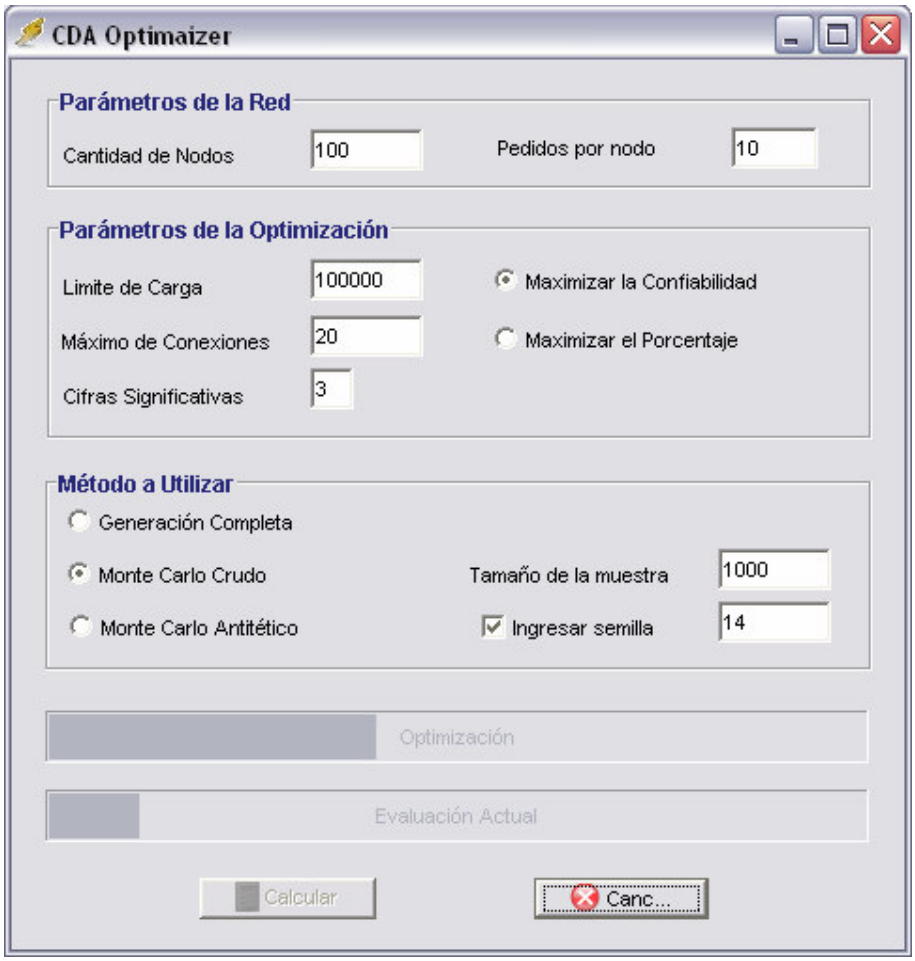


Fig. 5: Snapshot del CDA Optimizer

Planteado desde el punto de vista matemático, el problema a resolver consiste en encontrar la pareja de valores de *D* (tiempo de vida máximo permitido a los paquetes que circulan en la red) y *CCP* (cantidad de conexiones promedio permitidas a cada nodo) que es solución del siguiente problema:

$$\begin{cases} Max\ CDA \\ S.A.\ C < L \end{cases}$$

en donde CDA se refiere a la confiabilidad diámetro acotada de la red óptima, y *C* hace alusión a la función de carga que definimos en secciones anteriores de este documento.
El espacio de problemas sobre el que buscaremos el óptimo queda definido como una matriz de
N-2 x *N*-1, en donde consideraremos *D* variando en las abscisas, y *CCP* en las ordenadas. Cada punto (*CCP*, *D*) de este espacio de problemas, definirá, junto con los parámetros *N* y *CPN*, un grafo genérico de la forma de los grafos definidos como entrada en la herramienta de evaluación.

El valor de CCP_{max} ingresado, define un punto de corte en la matriz de problemas que definimos que nos permite eliminar del espacio de búsqueda todos los puntos con $CCP > CCP_{max}$. Esta restricción refleja una situación común en los sistemas de redes reales, en los que no es posible definir valores arbitrarios para la cantidad de conexiones permitidas a cada nodo cliente de la red.

A nivel de diseño e implementación de la aplicación, encararemos la solución del problema planteado en términos del algoritmo que ideamos y presentamos resumidamente en la sección 3.4.
Esto divide el algoritmo de optimización en dos etapas claramente identificables:

1. Definición del conjunto *F*, definido en la sección 3.4, de puntos del espacio de problemas que son límite en cuanto al cumplimiento de la restricción de carga indicada.
2. Búsqueda entres los puntos de *F* de aquel de ellos que maximiza el valor de la función de confiabilidad diámetro acotada, y retorno del mismo como valor óptimo para el problema presentado.

Para la primera etapa del problema, teniendo en cuenta los problemas de performance de la función de evaluación de la carga, se utilizará en su lugar la fórmula analítica presentada en la sección 3.2.3. Más aun, a partir de la misma y teniendo como entrada el valor límite que permitiremos para la función de carga que se recibe como parámetro de entrada, podremos despejar el valor de *D* límite para cada valor de *CCP* de manera trivial.
Para la segunda parte del algoritmo, la implementación es igualmente sencilla, bastando recorrer el conjunto de puntos incluidos en la frontera de carga, aplicando la evaluación de la confiabilidad de cada uno y guardando la información de aquel punto que maximiza este valor para ser retornado como el óptimo para el problema planteado.

Construida la herramienta de optimización, se realizaron algunas pruebas para la obtención de datos en pos de un estudio del comportamiento de la solución ante parámetros de entrada de diferentes características. Algunas conclusiones que se obtuvieron producto de estas pruebas son:

- Existe un óptimo trivial al problema que se encuentra en la posición (*CCP*=*N*-1, *D*=1) de nuestra matriz de problemas. En la práctica este valor corresponde a una red en la que todos los nodos tienen un enlace de confiabilidad 1 (confiabilidad = *CCP*/*N*-1) con el resto de los nodos de la red, cosa bastante poco probable en una red mediana a grande real. Por esta razón en nuestra matriz de problemas *CCP* varía entre 1 y *N*-2.

- Independientemente de lo mencionado en el punto anterior, es notorio a partir de los resultados de las pruebas que el los valores óptimos tienden a estar en la zona de combinaciones de valores altos de *CCP* y valores bajos de *D*.
  Que el óptimo se encuentre en la zona indicada se justifica por el hecho de que para valores grandes de *CCP* los estados que se obtienen para el grafo de cuestión son en su mayoría completos o casi completos por lo que alcanza con valores chicos de *D* para que haya conectividad.
  En cambio, para la zona en que *CCP* toma valores chicos, por más que *D* tome valores elevados, los estados que se generen van a tener pocas aristas lo que aumenta la probabilidad de que el estado que se genere no sea conexo y para los que la confiabilidad es 0.
  Esta comprobación empírica ameritaría un estudio analítico en busca de las causas de este comportamiento de las soluciones, lo cual se escapa al alcance de los objetivos de este proyecto, quedando como una puerta abierta para la realización de trabajos futuros.

- En los casos en que se fija un valor para el máximo *CCP* posible -CCP_{max}-, se observa por lo general que el óptimo calculado se encuentra en (CCP_{max}, D_0), en donde D_0 está dado por el menor valor de *D* para el que se cumple la restricción de carga, o en el "primer escalón siguiente de la frontera".

- Puede llegar a suceder que todo par (*CCP*, *D*) que cumpla la restricción de carga tenga CDA=0, en cuyo caso no existe un valor optimo para esa restricción de carga. Esto sucede cuando el límite de carga ingresado no es lo suficientemente grande en relación al tamaño del grafo indicado.

- Para la frontera de carga de nuestra matriz de cargas, se comprueba el comportamiento esperado con una velocidad de crecimiento lineal en *D* y logarítmica en *CCP*, resultados

estos que se conocían de antemano a partir del análisis de la fórmula analítica usada para la construcción de la matriz.

- También con respecto a la frontera de carga, se observó un comportamiento interesante en la misma que es que si *N* y el límite de carga varían en forma proporcional, la forma de la frontera se mantiene. Este comportamiento, en principio, no se explica de forma intuitiva

- Análogamente a la matriz de carga construida, se podría construir una matriz de confiabilidades. Esta construcción no se ha realizado en esta primera instancia, ya que su construcción es muy costosa en tiempo de ejecución.

7. Conclusiones y trabajos futuros

7.1. Conclusiones

A lo largo de las diferentes etapas en las que dividimos el proyecto hemos ido recolectando resultados y estudios que nos han permitido ir obteniendo conclusiones de interés en cada una de ellas.

De la primera parte del proyecto, encarada para la definición del lenguaje de representación de grafos que utilizaremos a lo largo del mismo y para el desarrollo de las aplicaciones de traducción y visualización de grafos, queda el estudio y discusión realizada precisamente sobre lenguajes para representación de estructuras de grafos la cual puede servir de guía al momento de tomar una decisión de este estilo en otro tipo de trabajos.
La herramienta de traducción GXL-HEIDI hace su aporte en el sentido de la compatibilidad del resto de nuestros desarrollos con trabajos anteriores en el tema, dejando abierta la posibilidad también de extender esta interoperabilidad con otros sistemas que manejen otros tipos de representaciones de grafos.

Teniendo en mente que el objetivo central del proyecto es el estudio, tanto teórico como práctico, de los algoritmos de evaluación y optimización, las conclusiones y aportes más importantes que deja el trabajo efectuado surgen en estas etapas.
Resumimos a continuación algunos puntos referentes precisamente a los resultados obtenidos durante la evaluación y la optimización:

- El método GCE nos brinda la posibilidad de evaluar las funciones de interés definidas de forma exacta, aunque su costo (en tiempo de ejecución y consumo de recursos del sistema) lo hacen inviable para un estudio de grafos de tamaños no despreciables. El principal aporte de este método es servir como base para una validación posterior de los métodos no exactos implementados.

- Los métodos de Monte Carlo probaron una vez más ser robustos, sencillos y eficientes al momento de implementar simulaciones estadísticas que permitan resolver problemas que son inviables de resolver por métodos exactos, aunque aparece como necesaria la utilización de algún tipo de medida paliativa para el control de la varianza de los resultados que se obtienen por el MCC.

- El MCA implementado precisamente para lograr resultados de menor varianza que el MCC, registra resultados relativos en este aspecto salvo para el caso de la función de carga, para la que se muestra una mejora significativa en el sentido de la reducción de la varianza. Al momento de la considerar el mismo como una alternativa para la evaluación mediante simulación, es de tener en cuenta también el gran overhead que introduce la utilización del mismo en cuanto a los tiempos de ejecución, por lo menos en nuestro tipo de problemas.

- Sobre las funciones estudiadas y posteriormente implementadas para ser evaluadas, todas ellas mostraron los resultados que se podían esperar intuitivamente.
  En particular, en lo que refiere a la medida de confiabilidad diámetro acotada, la misma finalmente se estudió desde dos puntos de vista: a nivel global mediante la función de confiabilidad diámetro acotada RDA_V, y desde un punto de vista más particular como el promedio de las confiabilidades fuente-terminal para todos los pares de nodos del grafo objetivo mediante la función de porcentaje *P*. Las conclusiones que se sacaron en ambos casos muestran una relevancia importante del parámetro de tiempo de vida de los paquetes -*D*- sobre el valor de las funciones, existiendo un valor o zona de valores antes de la cual la confiabilidad es muy

cercana a 0, y superada la misma ésta toma un valor muy cercano a 1. Un comportamiento similar se puede observar para el parámetro de cantidad de conexiones permitidas a cada nodo en promedio -*CCP*-, aunque en este caso la velocidad del cambio es menor.
La función restante, la carga en la red medida como la cantidad de paquetes enviados por todos los nodos, presenta un comportamiento francamente creciente ante el aumento de cualquiera de los tres parámetros considerados, reiterándose también el aumento abrupto, que en este caso parece ser exponencial, en los valores para cambios pequeños en *D* o en *CCP*.

- Comparando los resultados obtenidos, en cuanto a tiempos de ejecución para los tres métodos, en las implementaciones en Java y C++, no se observa un comportamiento general consistente en la superioridad de uno de los métodos contra el otro. Lo que parece pasar es que para grafos chicos (del orden de los 100 nodos) Java produce menores tiempos de ejecución que C++, aunque esta tendencia se revierte para grafos medianos a grandes. En ambos casos las diferencias no son de gran significación, rondando el orden del 10%.

- La etapa de optimización fue la más rica en el sentido de conclusiones y aportes novedosos surgidos de la misma:
    - Se ideó, implementó y testeó un algoritmo específico para el problema particular que nos atañe. A pesar de que el mismo se basa algunas particularidades de nuestro problema, puede ser de utilidad para otros trabajos que encaren optimización de redes en los mismos parámetros que en nuestro caso.
    - Se observó que, si no se limita la cantidad de conexiones máximas permitidas para cada nodo, la pareja óptima de valores de *CCP* y *D*, tiende a estar en la zona de valores altos de *CCP* y bajos de *D*. La justificación de este comportamiento se detalla en la sección anterior del presente documento.
    - El óptimo en cuanto la maximización de la confiabilidad puede no ser único debido a que el método de evaluación de la CDA utilizado no es exacto, en cuyo caso nuestra implementación retorna aquel punto que maximice la confiabilidad y tenga mayor valor de *CCP*.
    - Además de retornarse el óptimo, se incluye en la salida del programa el conjunto de valores de la frontera de carga.

7.2. Trabajos futuros

Quedaron fuera de este proyecto algunas actividades que surgieron, muchas de ellas, como consecuencia de estas mismas conclusiones y el análisis de resultados asociado. De manera similar a lo que se planteó para las conclusiones, se presentarán los trabajos que quedan abiertos para el futuro para cada de las etapas o fases por la que atravesó el proyecto.

1. Representación de grafos y traductor: Este punto se tocó en una buena amplitud (cantidad de lenguajes considerados en el estudio) y profundidad a través de un documento específico referente al lenguaje elegido GXL. Como posible extensión podríamos plantear la inclusión de más lenguajes en el estudio comparativo. A nivel de implementación de la herramienta de traducción, queda abierta la tarea de diseñar y desarrollar un archivo DTD para la validación de la estructura del grafo GXL leído o creado, cosa que permite el lenguaje pero de la que nosotros no hacemos uso.

2. Visualizador: Dado el tamaño y la relativa complejidad de la herramienta de visualización, ésta presenta varios puntos plausibles de extensión para una futura versión de la misma:

- Se podrían considerar otros tipos de formatos de entrada para la información de los grafos, como puede ser HEIDI.
- Pasar de una herramienta de visualización a una de construcción y modificación de grafos.
- Permitir la visualización de más de un grafo al mismo tiempo, en diferentes ventanas internas de la aplicación por ejemplo.
- Implementar otros algoritmos de ordenamiento de los elementos del grafo en la ventana.
- Incluir los atributos de nodos y aristas tanto en el algoritmo de ordenación inicial como cuando se guardan las posiciones de los elementos del grafo.
- Al momento de realizar la evaluación de las funciones de interés sobre el grafo que se está visualizando, se podría tomar la cantidad de pedidos que recibe cada nodo de un atributo de los mismos de la misma forma que se hace con la confiabilidad de las aristas, en lugar de tomar el mismo como un parámetro global del grafo (*CPN*)
- Aplica aquí también el hecho de poder contar con un archivo DTD para la validación del archivo GXL de entrada.

3. Evaluador: Para comenzar, una ampliación interesante de la herramienta sería la inclusión de nuevas funciones de evaluación. Más aun, se podría llegar a pensar en mejorar la misma agregando algún mecanismo dinámico para la inclusión de las funciones.
En otro sentido, podríamos cambiar la interfaz de entrada y salida de datos de forma de tomar un conjunto de casos (cada caso sería un juego de parámetros), evaluar cada uno y luego retornar el conjunto de resultados obtenidos para los mismos. Esto simplificaría la realización de pruebas y el análisis de los resultados, y su implementación se podría encarar utilizando archivos de entrada y salida en algún formato (por ej., usando XML) predefinido. Referente también a la interfaz, también se podría cambiar la misma para la consideración de grafos no completos a través, por ej., de archivos GXL.
Respecto al análisis de los resultados que se obtuvieron, quedan algunas puertas abiertas para la continuación del trabajo sobre los mismos, particularmente en las causas de algunos comportamientos observados, como ser las diferencias de performance entre las evaluaciones en Java y C++ al variar el número de nodos de los grafos, o las varianzas elevadas en algunos casos de prueba usando el MCA, entre otros.
Sería interesante también, estudiar las causas de los cambios abruptos en las funciones de confiabilidad y porcentaje ante cambios pequeños en los valores de *D* o *CPN*.

4. Optimizador: Por último, en lo que respecta a la herramienta de optimización el principal frente de estudio parece ser la búsqueda de una explicación formal para el justificar que los óptimos se encuentren en la zona de valores altos de *CCP* y bajos de *D*.
En lo referente a la aplicación en sí, parece una alternativa interesante de expansión de la misma la utilización de alguna interfaz gráfica de salida que permita la visualización del espacio completo de problemas y de los valores de carga y confiabilidad calculados para cada uno de ellos.
Manteniendo el formato actual de retorno de la frontera de carga en un archivo de texto plano, sería práctico incluir la posibilidad de elegir la carpeta y el nombre del mismo.
Teniendo en mente que luego de las pruebas realizadas se conoce mejor el comportamiento de las soluciones, se podría mejorar el algoritmo de optimización para que se concentre en las zonas del espacio de problemas en donde tiende a estar el óptimo.

8. Referencias

[BPSMcQ 98]
Extensible Markup Language (XML) 1.0
World Wide Web Consortium, (T. Bray, J. Paoli, C.M. Sperberg, McQueen),
http://www.w3.org/TR/REC-xml, Febrero de 1998.

[CE 98]
"Series-parallel reductions in Monte Carlo network reliability evaluation"
H. Cancela y M. El Khadiri.
IEEE Transactions on Reliability, 47(2):159-164, 1998.

[CP 01]
Diameter constrained network reliability: exact evaluation by factorization and bounds.
H. Cancela y L. Petingi
Proceedings of ICIL'2001 (International Conference on Industrial Logistics), páginas 359-366, Okinawa, Japan, 9-12 de Julio de 2001.

[CP 04]
Reliability of Communication networks with delay constraints: computational complexity and complete topologies.
H. Cancela y L. Petingi
International Journal of Mathematics and Mathematical Sciences, 2004(29-32): 1551-1562, 2004.

[CU 93]
"HEIDI – Una herramienta de apoyo para el diseño de redes de comunicación"
H. Cancela, M. Urquhart.
Technical report INCO 93-01, 1993.

[EKMR 91]
Parallel estimation of 2-terminal network reliability by a crude Monte Carlo technique.
M. El Khaidiri, R. Marie y G. Rubino.
Sixth Internacional Symposium on Computer and Information Sciences, Antalaya, Turquía - Ocubre de 1991.

[EKR 92]
A Monte Carlo method based on antithetic variates for network reliability computations.
Mohamed El Khaidiri y Gerardo Rubino.
Publicación interna, IRISA, 1992.

[GPG 02]
GraphML Specification.
GraphML Project Group.
http://graphml.graphdrawing.org/, 2002.

[HSEW 02]
Graph Exchange Lenguaje.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL, Julio de 2002.

[JGp 00]
JGraph Ltda.
http://www.jgraph.com/
http://sourceforge.net/projects/jgraph/

[Maut 00]
Proyecto de Taller V: Método RVR en la simulación de medidas de confiabilidad en redes.
Antonio Mauttone.
InCo, Facultad de Ingeniería, UdelaR, Uruguay - Febrero de 2000.

[PIS 02]
Proyecto de Ingeniería de Software: Herramienta 'Sim Engine'
Varios autores. Tutor: Regina Motz.
Informe interno, Facultad de Ingeniería, Universidad de la República, Uruguay, 2002.

[PIS 03]
Proyecto de Ingeniería de Software: Sistema editor de metadatas 'EDM5'
Varios autores. Tutor: Fernando Carpani
Informe interno, Facultad de Ingeniería, Universidad de la República, Uruguay, 2003.

[PU 96]
Algorithms for the computation of communication network vulnerability indexes
L. Petingi, M. Urquhart.
Reporte Técnico InCo 96.04, InCo, Facultad de Ingeniería, UdelaR, Uruguay - 1996

[SDA 00]
http://www.dc.teknowledge.com/external/sda-graph/

:: Apéndice A

:: Lenguajes de representación de grafos basados en XML

A.1. Introducción

Por lo general, en gran parte de los trabajos que involucran investigación y desarrollo de aplicaciones sobre redes, es de vital importancia la elección de una estructura de datos adecuada para la representación de las mismas. Una forma más que natural de representar las redes, es a través de un grafo (ya sea dirigido o no) en el que los nodos representen las terminales de la red y las aristas las vías de comunicación entre las mismas.
En este punto, el problema que planteamos en el presente trabajo, es la búsqueda de un modelo de datos que nos permita representar grafos cumpliendo con algunos criterios de calidad que definiremos en su momento.
Un modelo de datos, definido de forma simple, es una herramienta que nos permite representar la realidad. La amplitud de esta definición nos obliga a definir algunos criterios de calidad, de forma de poder restringir el abanico de opciones que consideraremos para el caso particular de la búsqueda de un modelo de datos para la representación de grafos.

El objetivo del presente trabajo es realizar una breve presentación de algunas de las herramientas basadas en XML [BPSMcQ 98] que se encuentran disponibles para la representación de grafos, así también como la posterior comparación de algunas de estas herramientas, de forma de poder finalmente tomar una decisión sobre la elección del modelo de datos a utilizar en el desarrollo del proyecto.
Como mencionamos arriba, nos restringiremos, para esta decisión, en modelos de datos basados en la herramienta XML. Esto se debe, principalmente, en el auge que tiene XML en la actualidad como herramienta de representación de datos de forma de independizar dicha representación de las aplicaciones que la utilizan. Además de esto, una mirada rápida de las principales herramientas de representación de grafos de la actualidad nos muestra que la gran mayoría están basadas en XML, lo que parece marcar una clara tendencia sobre el futuro de la representación de grafos.

Se asume que el lector está familiarizado con los conceptos básicos asociados a los grafos, así también como que posee algunas nociones básicas de XML.

En la siguiente sección se hace una breve presentación del estado actual de las herramientas de representación de grafos. A continuación presentaremos las principales ventajas y desventajas de algunas de estas herramientas; finalizando en la última sección con una comparación de las mismas y con la elección de uno de los modelos de datos analizados para ser usado en futuros trabajos.

A.2. Estudio de herramientas de representación de grafos basadas en XML.

No es nuestra intención realizar una búsqueda exhaustiva de modelos de datos para la representación, sino que en esta sección nos limitaremos a la presentación de aquellos que aparenten tener una mayor vigencia, una amplitud de uso importante, y una proyección de mantenimiento y desarrollo futuro interesante.

Históricamente la representación de grafos se inició con utilizando archivos de texto plano que definían niveles de anidamiento y nombres diferentes para cada uno de los objetos del grafo que se desearan representar. Dichas definiciones eran propias de cada proyecto, dificultando así el intercambio de grafos entre diferentes aplicaciones.
Un primer intento por definir un formato estándar de representación de grafos, surgió en 1997 bajo en nombre de GML (Graph Modelling Language) [Himsolt 97]. No está basado en XML, pero es un antecesor para la mayoría de las herramientas basadas en XML que surgieron más adelante.

Entrando al campo de las herramientas basadas en XML, muchas han surgido desde la aparición de XML. Muchas de ellas no tuvieron mayor difusión, distinguiéndose de entre ellas algunas que consideraremos en el presente documento.
La proliferación de formatos de XML para la representación, ha sido producto de la falta de una definición internacional sobre un estándar basado en XML con dicho propósito, como si ha surgido en otras áreas (IFX para las finanzas y MathML para las matemáticas, por nombrar dos solamente) En vista de esta realidad, nuestra búsqueda se centrará en aquellas herramientas que históricamente han pujado por convertirse en estándar en el área de la representación de grafos.
Luego de una rápida búsqueda de herramientas para representación de grafos basadas en XML, el conjunto de candidatos se reduce a tres: GXL, GraphML y XGMML.

A continuación presentaremos muy brevemente estas tres opciones y discutiremos su factibilidad para el problema que se nos plantea.
Los criterios que consideraremos para la evaluación de las herramientas son básicamente los siguientes:

- Simplicidad e intuitividad de uso.
- Posibilidad de representación de toda la gama de grafos que nos encontraremos en nuestro problema particular, especialmente representación de atributos en nodos y aristas.
- Acceso libre a los archivos de definición del lenguaje (XML-DTD o XML Schema)
- Existencia de herramientas que faciliten su uso e integración en otras aplicaciones a desarrollar.
- Existencia de una experiencia de uso y una comunidad de usuarios medianamente importante.
- Continuidad del trabajo de desarrollo de la herramienta.

A.2.1. Graph eXchange Lenguage: GXL

Esta herramienta, al igual que las otras, está definida como un subconjunto de XML limitado mediante la utilización de un documento XML-DTD o XML Schema.
GXL fue originalmente diseñado como una herramienta de intercambio de información entre aplicaciones [HSEW 02a]. Dado que diferentes aplicaciones pueden representar los datos de diferentes formas, los diseñadores de GXL decidieron que el formato mas adecuado para la representación de la información a intercambiar era un grafo.
A partir de esto GXL se ha hecho muy popular como lenguaje de representación de grafos independientemente de que su objetivo sea para el intercambio de información entre

aplicaciones, siendo utilizado en la actualidad en herramientas CASE y sistemas de transformación y visualización de grafos (por ej., en el sistema GTLX)

Entrando un poco más en lo que es la semántica de GXL, la misma prevé desde las construcciones básicas (nodos, aristas) hasta construcciones más complejas como hipergrafos. Dado que para nuestro problema nos basta con representación de grafos simples, de aquí en más nos concentraremos solo en las construcciones básicas del lenguaje.

Un documento GXL permite representar uno o más grafos. Cada grafo, a su vez, está formado por un conjunto de elementos que pueden ser de dos clases: nodos o aristas. Las aristas conectan siempre pares de elementos (o sea, no se pueden representar multiaristas) que pueden ser de cualquier clase (una arista puede conectar un nodo con otra arista, por ej.). A pesar de que una arista siempre tiene un nodo de origen y un nodo de destino diferenciados, la misma puede ser dirigida o no.
Para nuestro problema particular, una de las características más importantes que estamos buscando en la herramienta es la facilidad de representar atributos de los nodos y las aristas. En este sentido, GXL permite hacer esto de forma más que intuitiva, usando la misma construcción tanto para los nodos como para las aristas. Los atributos (al igual que el resto de los elementos) están identificados unívocamente por un identificador, además de tener un nombre, un tipo y un valor. Los tipos predefinidos por el lenguaje abarcan desde los básicos (voléanos, enteros, strings, etc.) hasta ennumerados; tipos compuestos (conjuntos, bags, secuencias, etc.) y URI's.

La utilización de esta herramienta parece sencilla e intuitiva en los casos de grafos simples (como los que encontraremos en nuestro espacio de problemas), y a simple vista parece tener construcciones para los tipos de grafos que manejaremos.

Otro aspecto que mencionamos considerar al momento de la elección de la herramienta, es la disponibilidad de aplicaciones, extensiones y plug-inns.
En este sentido es en el que más se destaca GXL, contando, entre otras cosas, con una biblioteca para el manejo de archivos GXL para Java y una aplicación (JGraphpad) para la edición de grafos y su posterior almacenamiento en formato GXL.

Por último, en lo que se refiere a la actualización del trabajo de desarrollo se nota que a pesar de que la especificación de la herramienta no ha cambiado en los últimos años, el trabajo de desarrollo y extensiones de la misma ha sido y sigue siendo bastante intenso, contándose por ej., con una última versión de la aplicación JGraphpad de este año. Esto es, aparentemente, fruto y consecuencia de que la herramienta cuenta con una gran cantidad de adeptos y usuarios en todo el mundo.

A.2.2. GraphML.

La segunda herramienta que mencionaremos surgió como una extensión de una herramienta previa denominada GraphXML, la que a su vez fue presentada en el evento 'Graph Drawing 2000' [GPG 02].
La herramienta en sí está formada por una capa estructural que define el subconjunto de XML que se maneja, a través del denominado GraphML Schema (XML Schema para la definición del sublenguaje manejado por el GraphML), contándose también con una especificación a través del GraphML DTD.
Además de esta capa estructural, cuenta con dos extensiones que permiten definir atributos e información de parseo.

Un grafo para el modelo de datos definido por GraphML se denomina Multigrafo Mixto Etiquetado y puede contener construcciones complejas como loops y multiaristas.
Como en el caso anterior, en este análisis nos centraremos en las construcciones que presumiblemente serán de utilidad para nuestro problema en particular.

En este sentido, la herramienta define construcciones básicas para nodos y aristas, con la particularidad de que el grafo podrá contener tanto aristas dirigidas como no dirigidas y pudiéndose especificar uno de estos valores como valor por defecto a nivel del grafo. Al igual que en GXL, tanto si la arista es dirigida o no la misma tiene definidos el nodo origen y el nodo destino de la misma.
Una construcción nueva que aparece en esta herramienta, es la posibilidad de definir puertos en los nodos (un puerto es punto de entrada o salida en el que puede incidir una de las aristas incidentes en el nodo), la cual podría ser de utilidad en nuestro problema particular.

En el área de la definición de atributos para los elementos del grafo, como se mencionó antes GraphML define los mismos en una extensión de la estructura de definición del grafo propiamente dicho.
A su vez, dentro de esta extensión, es posible definir tres tipos de atributos:

- Datos no estructurados: función parcial que asigna valores dentro de un rango a elementos del grafo.
- Datos estructurados: se definen de forma similar a los datos no estructurados, redefiniendo además el tipo complejo 'data-extension.type'.
- Datos tipeados: definición de tipos estructurados para su asignación a los atributos.

En esta área aparentemente la tarea de definir atributos para los elementos del grafo no es muy sencilla ni intuitiva. El hecho de que se halla intentado abarcar muchas posibilidades ha provocado un aumento de la complejidad para la definición de atributos sencillos. Este es uno de los principales factores negativos que nosotros consideramos en la herramienta GraphML.

Pasando a considerar aplicaciones auxiliares y extensiones de la herramienta, las mismas son muy pocas y de muy baja calidad.
A su vez, los desarrolladores de la herramienta (GraphML Project Group) han prometido a través de su página de Internet [GPG 02] más documentación sobre la herramienta, así también como una extensión de la misma que facilite su integración con GXL. Ambas cosas no están disponibles aun, lo que aparentemente estaría marcando la discontinuidad de los trabajos sobre la herramienta.

A.2.3. Extensible Graph Markup and Modeling Language: XGMML.

La última herramienta que se estudió como factible para ser usada a lo largo del proyecto es el XGMML [PK 01]. La misma está basada en GML (Graph Markup Language), una herramienta para representación de grafos no basada en XML de amplio uso.
Como en los casos anteriores, XGMML es un sublenguaje de XML definido a través de un XGMML Schema. Originalmente fue pensado como un formato de intercambio de grafos entre clientes y servidores en un ambiente Web, pero su uso se ha extendido para la representación de grafos en casi cualquier contexto.

XGMML respeta el formato de GML, agregándole las ventajas de interoperabilidad que presenta XML y la posibilidad de incluir información extra y arbitraria sobre los elementos representados en el grafo.
Un documento XGMML define la estructura de un único grafo, permitiendo que el mismo contenga estructuras complejas (las cuales serán nuevamente obviadas).
Los elementos básicos del grafo son representados a través de construcciones simples para los nodos y las aristas. El grafo puede ser dirigido o no, pero esta propiedad está definida a nivel del grafo y no a nivel de las aristas, por lo que no es posible representar aristas dirigidas y no dirigidas dentro de un mismo grafo.
Una diferencia con respecto a los otros lenguajes estudiados, es la posibilidad de definir información sobre la representación gráfica del grafo junto con la información del grafo propiamente dicha. Dado que esta información es opcional, la misma no presentaría inconveniente para nuestro caso particular (en el que pretendemos usar la herramienta como

modelo de datos para los grafos, manejando la representación gráfica de los mismos de forma independiente de los datos en sí), bastándonos simplemente con obviar la misma.
Otra construcción particular de esta herramienta, es que tanto las aristas como los nodos tienen definidos un atributo por defecto (weight) que representaría el peso de cada una de ellas. En nuestro caso, dicho atributo no es de mayor utilidad, dado que pretendemos usar una estructura que permita la definición de una cantidad arbitraria de atributos para cada elemento. Nuevamente, esto no presenta un inconveniente mayor, dado que el atributo weight es opcional tanto en aristas como en nodos.

Continuando con el tema de la definición de atributos a los elementos del grafo, la forma en que XGMML permite esto, es a través de la construcción 'att'. Los atributos definidos de esta manera pueden ser atributos simples (formados por un nombre, un tipo y un valor), o atributos muy complejos definidos usando otras herramientas como XML o RDF [MMMcB 03] (en caso de que el grafo XGMML se encuentre embebido en un documento XML o RDF).
La definición de atributos simples, es asimismo simple e intuitiva, teniendo además la opción de definir atributos complejos, aunque en estos casos se introduce una dependencia con el archivo externo que debe contener al grafo.

En lo que se refiere a las extensiones o aplicaciones disponibles para esta herramienta, no se ha detectado la existencia de alguna (más que de conversores GML-XGMML y XGMML-GML) ni se ha detectado intención de desarrollos en este aspecto por parte de los responsables de la herramienta.
Tampoco se ha palpado la existencia de una comunidad de usuarios considerables, ni de esfuerzos de depuración o mantenimiento de la herramienta.

A.3. Comparación de las Herramientas y selección de una

A lo largo de las tres secciones anteriores se han discutido muy brevemente y en los aspectos que nosotros consideramos más importantes para nuestro proyecto particular, tres herramientas para la representación de grafos basadas en XML.

De los puntos mencionados y de algunos otros, hemos llegado a la decisión de elegir GXL como modelo de datos para los grafos en nuestro proyecto.
Esta elección se ha basado en aspectos externos a lo que es cada uno de los lenguajes considerados en sí mismo, ya que los tres son fáciles de entender e intuitivos en su uso para los tipos de grafos que se planean manejar. De hecho esta elección se fundamenta básicamente en los siguientes factores:

- Existencia de una amplia variedad de herramientas específicas para GXL de distintos tipos (bibliotecas Java, aplicaciones para la edición de grafos, etc.)
- Continuidad de los esfuerzos de mejora y desarrollo en torno al lenguaje hasta la fecha.
- Existencia (aparente) de una importante comunidad de usuarios de la herramienta, que se ve plasmada en la existencia de listas de correos (http://mailhost.uni-koblenz.de/mailman/listinfo/gxl) donde se da soporte sobre la herramienta. No se detectaron comunidades de este estilo para ninguna de las otras dos herramientas estudiadas.
- Facilidad en la representación de atributos de grafos y aristas: a pesar de que los tres lenguajes definen construcciones de este tipo, GXL es el que, a nuestro entender, lo hace de forma más sencilla e intuitiva.

A.4. Referencias

[BPSMcQ 98]
Extensible Markup Language (XML) 1.0
World Wide Web Consortium, (T. Bray, J. Paoli, C.M. Sperberg, McQueen),
http://www.w3.org/TR/REC-xml, Febrero de 1998.

[GPG 02]
GraphML Specification.
GraphML Project Group.
http://graphml.graphdrawing.org/, 2002.

[Himsolt 97]
GML - Graph Modelling Language.
Michael Himsolt, Universität Passau, Passau, Alemania.
http://infosun.fmi.uni-passau.de/Graphlet/GML/, 1997.

[HSEW 02a]
Graph Exchange Lenguaje, Background.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/Introduction/Backgound.html, Julio de 2002.

[HSEW 02b]
GXL 1.0 – XML Schema definition.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/dtd/gxl-1.0.html, Julio de 2002.

[MMMcB 03]
RDF Primer.
W3C Working Draft: Frank Manola, Eric Miller y Brian McBride.
http://www.w3.org/TR/rdf-primer/, Octubre de 2003.

[PK 01]
XGMML 1.0 Draft Specification.
John Punin y Mukkai Krishnamoorthy.
http://www.cs.rpi.edu/~puninj/XGMML/draft-xgmml.html, Junio de 2001.

[Rodríguez 02]
Lenguajes canónicos para la descripción de grafos: estudio y transformación entre esquemas de distintos modelos de datos
Pablo Rodríguez.
InCo, Setiembre de 2002.

[TBMBM 00]
XML Schema.
H. Thompson, D. Beech, M. Maloney, P. Biron y A. Malhotra
N. Mendelsohn editors, URL: http://www.w3.org/TR/xslt, Febrero de 2000.

:: Apéndice B

:: Graph Exchange Language (GXL)

B.1. Introducción

El presente informe es una introducción a la herramienta para representación de grafos GXL. Según sus propios creadores, "...GXL es un formato estándar basado en XML [BPSMcQ 98] para compartir datos entre diferentes herramientas" [HW 00].

De aquí surgen algunos hechos interesantes respecto de GXL.
El primero es que GXL es un sublenguaje de XML, por lo que hereda toda la potencialidad de esta herramienta de modelado, contándose además con una experiencia interesante en el manejo de XML que puede ser volcada a la manipulación de documentos GXL.
Por otro lado, y muy ligado a lo anterior, GXL fue desarrollado teniendo en mente la idea de pudiera volverse con el correr de los años, en un estándar mundial para la representación de grafos. Esto ha sido potenciado por sus creadores al intentar obtener la mayor retroalimentación posible de parte de los usuarios de GXL, manteniendo la sintaxis de GXL como un estándar abierto, y usando el ya estandarizado XML como herramienta base en el desarrollo de GXL.
Por último, GXL nace como una herramienta para permitir la interoperabilidad entre herramientas de reingeniería de software, eligiéndose el formato de grafos debido a su flexibilidad y adaptabilidad en este ambiente, además de su independencia de representación con respecto a la herramienta en que se utilice.

En las siguientes secciones ampliaremos estos puntos que hemos mencionado muy brevemente aquí, además de presentar y comentar la sintaxis de GXL, acompañada de algunos ejemplos.
En la sección B.2 brindaremos una muy breve historia del nacimiento de GXL, sus orígenes y su desarrollo hasta el día de hoy; continuando en la sección B.3 con una presentación de la sintaxis de GXL. En la sección B.4 se presentarán algunos puntos de desarrollo futuro de la herramienta; mientras que en la sección B.5 se describirán algunas herramientas, bibliotecas y plug-ins para el manejo de GXL; culminando en la sección B.6 con una conclusión de lo que GXL nos brinda como herramienta de representación de grafos.

B.2. Orígenes de GXL

Como ya mencionamos GXL nace como una herramienta de representación de grafos en el ambiente de la reingeniería de software, como un desarrollo conjunto de varias universidades: University of Waterloo (Canadá), Darmstadt University of Technology (Alemania), University of California (EE.UU) y University Of Koblenz-Landau (Alemania).

La idea que motiva el desarrollo de la herramienta es la de definir un estándar internacional para el intercambio de información derivada de programas en forma de grafo, o mas generalmente para intercambiar información en forma de grafo [HW 00].

Desde el primer momento se planteó como un objetivo de diseño que el modelo se basara en estándares internacionales, por lo que no es de extrañar la elección de XML como lenguaje base de la herramienta.

La versión inicial de GXL fue el resultado de la mezcla de ideas basadas en tres herramientas de utilización en el ambiente de la reingeniería de software de la época:

1. El formato GraX (Graph Exchange Format) para el intercambio de grafos dirigidos con atributos [EKW 99].
2. El formato para representación de grafos TA (Tuple Attribute Language) [Holt 97].
3. El sistema PROGRES para la representación y reescritura de grafos.

A partir de esta base inicial, se fueron agregando ideas tomadas de otras aplicaciones y formatos para ampliar la capacidad de la herramienta. Entre ellas podemos los formatos de intercambio de información RPA (Relation Partiton Algebra) y RSF (Rigi Standard Format), y los formatos de representación de grafos GML, XGMML y GraphML [WRK 01].

Un primer paso en el camino de GXL en post de convertirse en estándar para el intercambio de información entre aplicaciones, fue dado en el 2001 al ser definido como estándar por el seminario “Interoperability of Reverse Engineering Tools” de Dagstuhl [WKR 01].
En este momento se logró redondear lo que sería la versión 1.0 de GXL. Esta fue la primera versión estable del lenguaje y fue la que logró la popularización de GXL como lenguaje de representación de grafos.
A partir de ese momento cada vez mas empresas (Bell Canada, IBM Centre Canada, Nokia Research Center en Finlandia y Phillips Research en Holanda, entre otras) y universidades (University of Berne en Suiza, University of Koblenz de Alemania, University of Oregon en EE.UU y University of Waterloo en Canadá, entre otras) han adoptado GXL como lenguaje de intercambio entre aplicaciones, tanto para fines investigativos como para proyectos reales.

La siguiente figura muestra esquemáticamente como fue el desarrollo del lenguaje en sus primeras etapas y las diferentes influencias que tuvo durante el proceso:

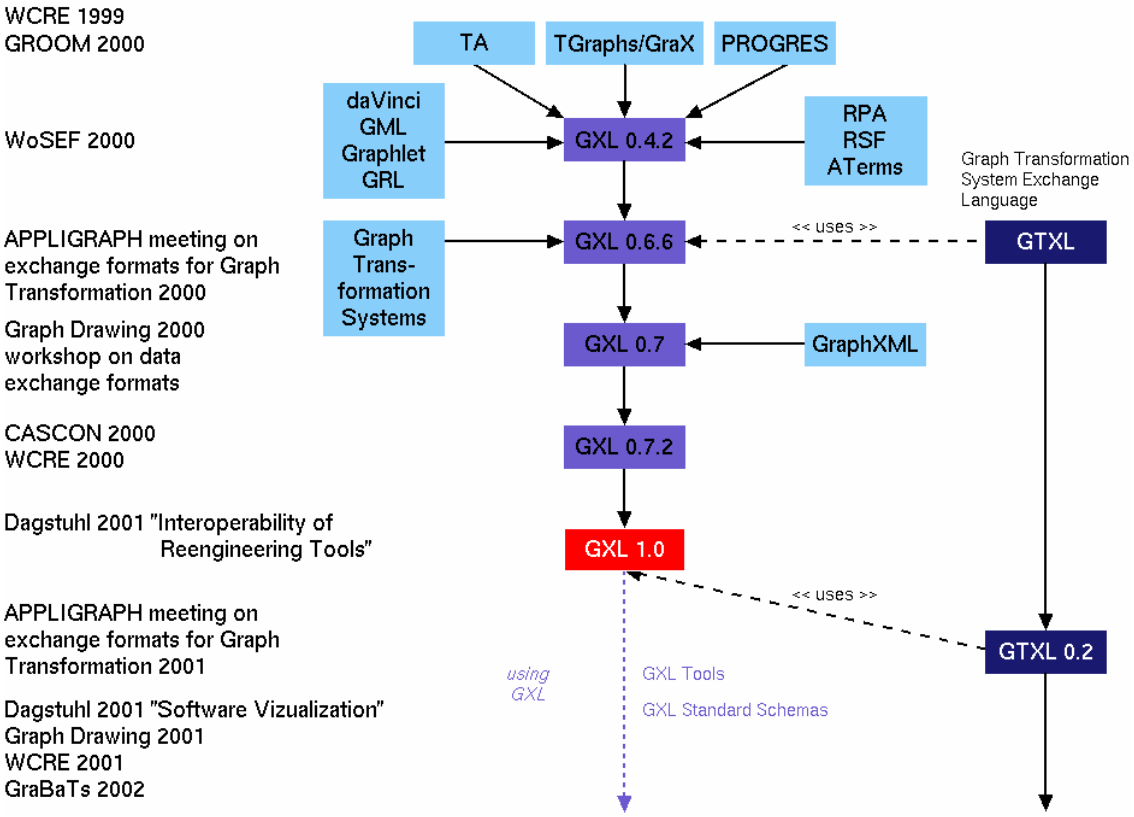


Fig. B.1: Influencias en el desarrollo de GXL

En el presente GXL ha sido adoptado ampliamente en el ambiente investigativo de la reingeniería de software y de la transformación de grafos, estando en consideración por otras comunidades.

B.3. Propiedades generales

Al tratarse de un sublenguaje de XML, es lógico pensar que la definición de GXL se encuentre dada mediante un DTD o un XML Schema. Los mismos se encuentran disponibles en [HSEW 02b] y [HSEW 02c] respectivamente.
Ambas definiciones son abiertas, lo que da al usuario la posibilidad de adaptar el lenguaje a sus necesidades específicas en caso de que así lo requiera.

GXL ofrece un soporte versátil para la representación muchos tipos de grafos, basándose en la representación de grafos tipeados, ordenados, dirigidos y que soportan el uso de atributos para cualquier elemento. Un grafo con estas características se denomina TGrafo.
Los TGrafos permiten la definición de atributos a nodos y aristas, y son tipeados. Cada tipo puede ser asignado a un atributo particular en el esquema. Los tipos que soporta GXL son los atómicos (Bool, Int, etc.), o compuestos (Bag, Set, Seq y Tup).
Los TGrafos también son ordenados, lo que implica que el conjunto de nodos, el conjunto de aristas del grafo, y el conjunto de aristas incidentes en un nodo, tienen definido un orden total. Este orden facilita la implementación de algoritmos determinísticos sobre el grafo (según [WKR 01], Pág 2)

Al definir los tipos de grafos que maneja el lenguaje, uno de los objetivos primordiales fue el de tener la capacidad de representar grafos complejos sin que esto aumente la complejidad en la representación de grafos simples. En este punto se logró un equilibrio interesante, soportándose construcciones complejas (como hipergrafos y grafos jerárquicos) sin que esto afecte la simplicidad de representación de grafos simples.

Otro punto interesante es que GXL soporta la representación del esquema de un grafo de la misma forma en que se representan los grafos. De esta forma podemos definir la estructura de un cierto grafo usando también el lenguaje GXL, lo que resulta importante para aplicaciones que intercambien grafos.

B.4. Sintaxis de GXL

En lo que se refiere a su sintaxis, GXL define una estructura de tags que deben cumplir los documentos GXL válidos. Dicha estructura queda definida a través de los documentos, ya sea DTD o XMLSchema, que definen la metaestructura de GXL.
Esta especificación de la sintaxis soportada por GXL, puede ser representada esquemáticamente, como muestra la figura a continuación:

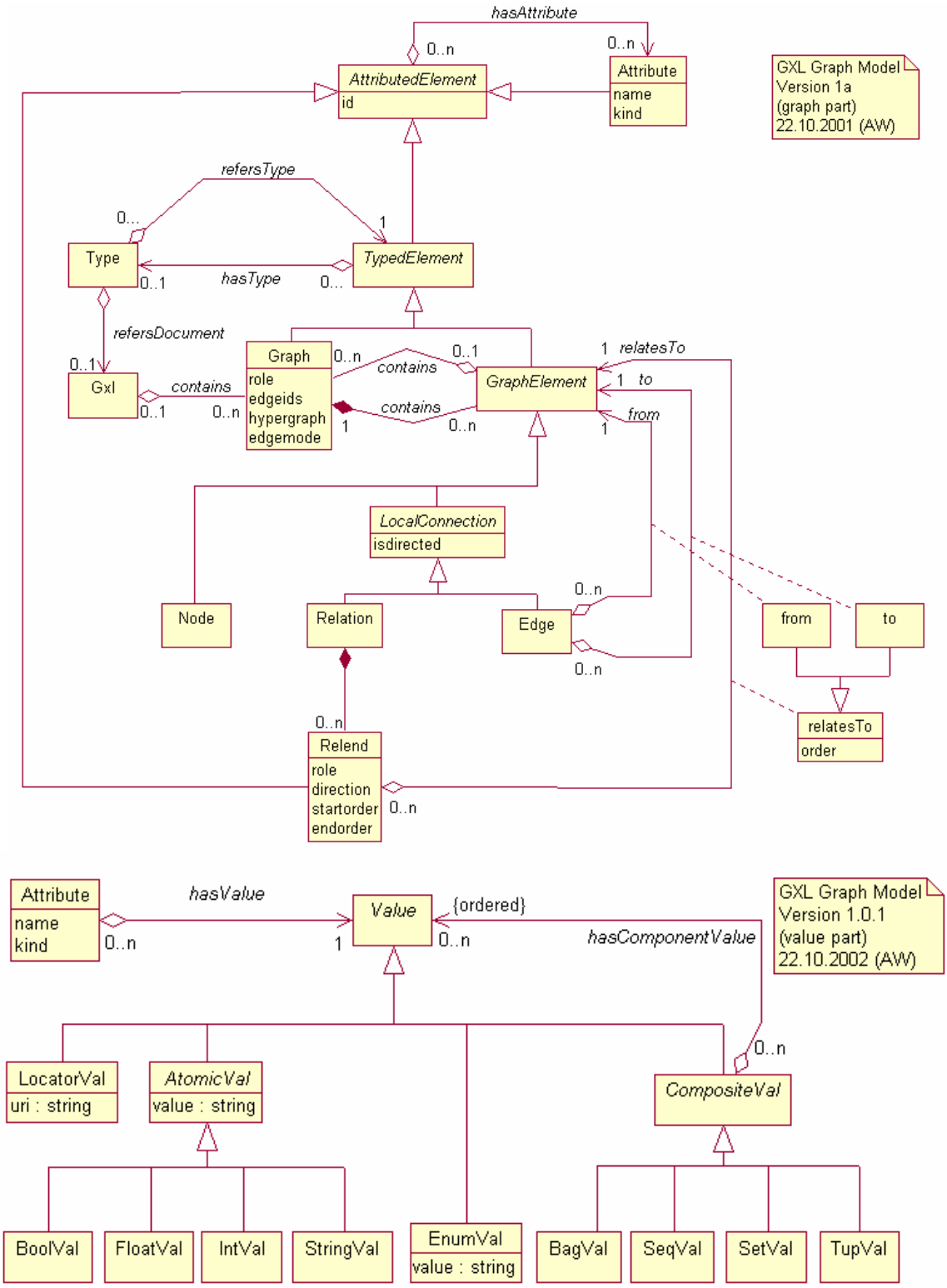


Fig. B.2: Sintaxis de GXL

Partiendo de este esquema, comentaremos brevemente los tags principales que define GXL, teniendo siempre en mente aquellos que nos serán útiles a lo largo de nuestro proyecto.

<GXL>

Es la raíz de cada archivo GXL. Un archivo GXL puede contener solo un elemento <GXL>. No tiene atributos definidos y puede contener más de un grafo, es más el único elemento que puede contener es el <graph>.

<graph>
Define un grafo dentro del documento. Tiene definidos 4 atributos, que mencionamos en la siguiente lista:

- id: Identificador del grafo dentro del documento.
- role: String que sirve para darle una descripción al grafo.
- edgeids: Valor booleano que indica si las aristas requieren identificadores o no; por defecto es False.
- hypergraph: Valor booleano que indica si el grafo es un hipergrafo (o sea, si soporta aristas n-arias); por defecto es False.
- edgemode: Indica si el grafo es dirigido o no, tomando los valores “directed” (por defecto) o “undirected” respectivamente. También puede tomar los valores “defaultdirected” (para indicar que las aristas pueden ser dirigidas o no, pero serán dirigidas por defecto) y “defaultundirected” (ídem tomando por defecto que las aristas son no dirigidas).

Dentro del elemento <graph> está permitida la aparición de cualquier cantidad de elementos <node>, <edge> y <rel>, sin importar tampoco el orden de aparición de los mismos. Un grafo puede hacer referencia al esquema que lo define, para lo que se utiliza el elemento opcional <type> junto con una URI del esquema. También se pueden declarar 0 o más atributos de usuario para el grafo mediante la utilización del elemento <attr>.

<node>
Elemento que representa la ocurrencia de un nodo, el cual tiene un solo atributo XML (id) correspondiente al identificador del nodo en el documento. Además de éste, <node> puede tener una cantidad no acotada de atributos definidos por el usuario, los que se incluirán como elementos <attr> asociados al nodo.
Otros elementos que pueden aparecer dentro de un elemento <node> son: <type> para indicar el tipo del nodo (los tipos pueden ser definidos por el usuario), y <graph> que permite incluir uno o mas nuevo grafos dentro del nodo, grafos estos que representan un nivel inferior en la jerarquía del grafo.

<edge>
Las aristas definidas por el tag <edge> son siempre binarias e independientemente de que sean dirigidas o no, tienen definidos un nodo origen y un nodo destino que se definen mediante atributos del elemento. La lista completa de atributos de <edge> es la que sigue:

- id: identificador de la arista en el documento.
- from: Identificador del nodo de origen de la arista.
- to: Identificador del nodo de destino de la arista.
- fromorder, toorder: En los grafos en que las aristas deben ser atravesada en un cierto orden, estos atributos son usados para almacenar el orden de incidencia de los nodos de origen y destino respectivamente; ambos son opcionales.
- isdirected: Valor booleano que indica si la arista es dirigida (True) o no (False).

Los identificadores usados para referenciar los nodos sobre los que incide la arista, deben corresponder a nodos definidos dentro del mismo grafo en que se está definiendo la arista.
Otra cuestión a resaltar es que el atributo ‘isdirected’ solo es tenido en cuenta cuando no entra en conflicto con el atributo ‘edgemode’ del grafo al que pertenece la arista.
Al igual que el nodo, la arista puede ser asociada a un tipo mediante la inclusión de un <type> asociado a la misma; pueden declararse atributos usando el elemento <element>; y pueden asociarse grafos en un nivel menor de jerarquía usando <graph>.

<attr>

Usado para declarar atributos indistintamente para grafos, nodos y aristas. Al igual que los demás elementos mencionados, <attr> tiene un atributo 'id' para la declaración de un identificador único del atributo, el cual está ligado al elemento que lo contiene. Para la identificación del atributo por parte del usuario, se utiliza el atributo 'name' del mismo, el cual debe ser único dentro del conjunto de <attr>s del elemento que contiene al <attr> en cuestión, y es obligatorio. Otro atributo que es posible definir dentro de <attr> es 'kind' cuya utilidad sería la de agrupar los <attr> de un elemento según clases definidas por el usuario siguiendo algún criterio particular.
Es posible definir atributos para un <attr>, lo que se consigue agregando <attr>s dentro del <attr> base.
La definición del valor del atributo se realiza mediante elementos definidos por el lenguaje, los que se agrupan en dos clases:

1. Valores atómicos:
    a) <bool>: Valores booleanos ("true" y "false" con minúsculas)
    b) <int>: Valores enteros
    c) <float>: Valores reales
    d) <string>: Strings de caracteres que siguen el estándar Unicode

2. Enumerados:
    a) <seq>: Secuencia de valores
    b) <set>: Conjunto de valores
    c) <bag>: Bolsa de valores
    d) <tup>: Tupla de valores

    Para todos los enumerados se cumple que los elementos que incluyen deben ser todos del mismo tipo, siendo los tipos cualquiera de los mencionados en 1. y 2.

Otros elementos definidos por GXL son <rel> y <relend> que representan aristas n-arias, y los "tentáculos" de una arista n-aria (o sea, un punto terminal de la arista) respectivamente. No entraremos en detalle en estos y otros elementos existentes, debido a que en la realidad de nuestro proyecto particular no son necesarios para el modelado de los grafos que manejaremos.

B.5. Herramientas

Un punto importante al momento de hablar de un lenguaje de modelado que se utilizará en aplicaciones implementadas en computadoras, es la disponibilidad de facilidades para el manejo del mismo en este ambiente.
En este punto GXL brinda una variedad interesante de posibilidades, algunas de ellas que mencionaremos brevemente a continuación. Para una lista completa de herramientas vinculadas a GXL, referirse a [HSEW 02d].

B.5.1. Biblioteca GXL para Java.

Una herramienta de gran utilidad para el desarrollo de aplicaciones Java que utilicen GXL es la biblioteca homónima. La misma define clases para representar todos los tags soportados por el lenguaje, junto con operaciones clásicas para el acceso a las propiedades de los mismos, además de operaciones que facilitan la lectura y escritura de archivos GXL.
Junto con la biblioteca, se pueden obtener las APIs de la biblioteca, documento muy útil para el programador al momento de escribir código utilizando la biblioteca.
Ambas cosas están disponibles de forma libre en [HSEW 02d].

B.5.2. JGraphpad.

Aplicación desarrollada en Java, bajo licencia GNU, para la edición de grafos y su posterior almacenamiento en formato GXL, así también como para la visualización de grafos almacenados en documentos GXL.
Tiene una interfaz bastante amigable, lo que facilita la creación de grafos relativamente complejos. Un gran punto en contra, es la imposibilidad de definir atributos para los nodos y aristas de los grafos especificados.

B.5.3. GXLValidator.

Ejecutable que recibe como entrada un documento en formato GXL y realiza la validación sintáctica del mismo, retornándose la lista de errores y sus ubicaciones dentro del documento en caso de que el mismo no sea válido.
Cuenta con opciones para chequear también la adecuación de un documento GXL a un esquema GXL determinado, entre otras opciones.
Está disponible de forma libre tanto para Windows (archivo exe) como para Linux (archivo sh).

B.6. Desarrollo futuro

Al momento de escribir este documento, al buscar referencias sobre el posible desarrollo futuro del lenguaje en la página oficial, solo encontramos una propuesta para el desarrollo de una versión 1.1, que además de no agregar cambios sustanciales al lenguaje, la misma es de Diciembre de 2002.
Esta versión fue generada a partir de sugerencias enviadas por los usuarios del lenguaje, y está presentada en la página como una versión alfa.
Dado el tiempo transcurrido desde la presentación de la versión alfa de GXL 1.1, el hecho de que no se tengan más novedades sobre la misma, aparenta ser un síntoma de que no tuvo mayor impacto en la comunidad de usuarios del lenguaje. Otro hecho que ratificaría esta idea, es que todas las herramientas que hacen soporte a GXL están basadas en su versión 1.0, no encontrándose ningún avance para la versión 1.1 en las mismas.

De todo esto podemos llegar a la conclusión de que la versión 1.0 se estabilizó como la versión definitiva de GXL. Los desarrollos futuros se centrarían entonces en el desarrollo de aplicaciones, bibliotecas o plug-inns que faciliten el desarrollo de aplicaciones que utilicen GXL para diferentes lenguajes y en diferentes plataformas.

B.7. Referencias

[BPSMcQ 98]
Extensible Markup Language (XML) 1.0
World Wide Web Consortium, (T. Bray, J. Paoli, C.M. Sperberg, McQueen),
http://www.w3.org/TR/REC-xml, Febrero de 1998.

[EKW 99]
GraX - An Interchange Format for Reengineering Tools.
J. Ebert, B. Kullbach, A. Winter.
Proceedings of the 6th Working Conference on Reverse Engineering (WCRE'99), S. 89-98

[Holt 97]
An Introduction to TA: The Tuple Attribute Language
Ric Holt.
Department of Computer Science, University of Waterloo and Toronto, 27 Feb. 1997.

[HSEW 02a]
Graph Exchange Lenguaje, Introduction.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/Introduction/Intro.html, Julio de 2002.

[HSEW 02b]
Graph Exchange Lenguaje, DTD.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/dtd/dtd.html, Abril de 2002.

[HSEW 02c]
Graph Exchange Lenguaje, XML Schema Definition.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/xmlschema/xmlschema.html, Julio de 2002.

[HSEW 02d]
Graph Exchange Lenguaje, Tool Catalogue.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/tools/tools.html, Julio de 2002.

[HSW 00]
Looking for a Graph Exchange Language.
Ric Holt, Andy Schürr, Andreas Winter.
APPLIGRAPH- Subgroup Meeting on Exchange Formats for Graph Transformation, Paderborn University, Alemania, Setiembre de 2000.

[HW 00]
A Short Introduction to the GXL Exchange Format.
Ric Holt y Andreas Winter.
Proceedings 7th Working Conference on Reverse Engineering (WCRE 2000), Panel on Reengineering Exchange Formats.

[WKR 01]
An Overview of the GXL Graph Exchange Language.
A. Winter, B. Kullbach, V. Riediger.
Springer Verlag: S. Diehl (ed.) Software Visualization · International Seminar Dagstuhl Castle, Germany, May 20-25, 2001 Revised Lectures.

:: Apéndice C

:: Herramienta de traducción HEIDI-GXL-HEIDI GXL Translator

C.1. Introducción

La aparición de XML [WWW 98] en los ruedos de la informática causó un impacto positivo en todas las áreas de la computación particularmente, propagándose luego a un conjunto de áreas en donde la computación era una herramienta fundamental o en donde empezaba a serlo. Esto se debió principalmente a su flexibilidad en la definición de marcas (tags) arbitrarias, y la posibilidad de utilizar documentos de validación de formato de los archivos XML (inicialmente fueron archivos DTD y más adelante XML Schemas).

Esta expansión del uso de XML en diferentes ámbitos, trajo aparejada la paulatina aparición de diferentes formatos, herederos de XML y compatibles con éste, con estructuras definidas para áreas específicas de la industria. Como ejemplos de estos sub-lenguajes podemos nombrar ebXML (Electronic Business XML Initiative) para el comercio electrónico, IFX (Financial Exchange) en el área financiera, y CML (Chemical Markup Language) en la industria química, entre otros.
Uno de estos sub-lenguajes basados en XML surgido en los últimos tiempos es GXL (Graph Exchange Language) cuyo objetivo es la representación de grafos [HSEW 02].

En este contexto de la representación de grafos, históricamente se han utilizado diferentes herramientas que han resultado eficaces y hasta eficientes, pero que debido al constante devenir de nuevas tecnologías con ideas renovadoras, han requerido su actualización y hasta su liso reemplazo.
En el marco de desarrollo de este proyecto, bajo la tutoría del Departamento de Investigación Operativa de la Facultad de Ingeniería de la UdelaR, se ha venido utilizando desde el año 1993 un lenguaje para la representación de grafos denominado HEIDI (Herramienta Inteligente para el Diseño de Redes de Comunicación Confiables) [CU 93]. A través de proyectos similares al nuestro se ha desarrollado una interfaz gráfica para la herramienta en el año 1995 y posteriormente, en el año 1996, se realizó una extensión del mismo.
De todas formas con el correr de los años, HEIDI ha entrado lentamente en desuso. Herramientas nuevas y más poderosas son requeridas para mantenerse en línea con el resto de las aplicaciones y sistemas.
Dadas las ventajas que ofrece XML (algunas de las cuales pueden repasarse en [Rodríguez 02]) y la gran difusión y uso que el mismo tiene, parece una alternativa más que interesante la consideración del mismo como sucesor de HEIDI como lenguaje de representación de grafos en el marco de nuestro proyecto. Diferentes sub-lenguajes de XML específicos para la representación de grafos fueron evaluados en un trabajo previo que se incluye como **Apéndice A** de este documento, eligiéndose finalmente el lenguaje GXL.

En este punto, como en cualquier proyecto de evolución de herramientas, la necesidad de mantener una cierta compatibilidad con las herramientas anteriormente existentes es uno de los requerimientos más relevantes. En nuestro caso, el objetivo es poder utilizar la batería de herramientas que soportan HEIDI desarrolladas a lo largo de estos años, para grafos generados en GXL; así también como poder utilizar las herramientas que se están desarrollando en el presente proyecto con los grafos existentes en formato HEIDI.
De esta forma se pretende lograr algún nivel de interoperabilidad entre las aplicaciones ya existentes, y las aplicaciones que se están desarrollando en la actualidad, sabiendo que esta forma de interacción, a través de los lenguajes de representación de grafos, es mucho menos que la ideal.
Con este enfoque nos planteamos la tarea de implementar una aplicación de traducción de archivos HEIDI a archivos GXL y viceversa.

En el presente documento pretendemos hacer una presentación de la herramienta de traducción desarrollada, haciendo énfasis en su diseño y en algunos aspectos que puedan resultar interesantes de su implementación y su funcionamiento.

En las siguientes secciones se presentarán los objetivos y el alcance de la aplicación, luego se describirá el Modelo Conceptual sobre el que se basa el sistema construido, continuándose con la descripción y explicación mediante diagramas de colaboración de las dos operaciones básicas de la herramienta (léase, la traducción de un archivo en formato HEIDI a un archivo en formato GXL y la operación inversa), siguiendo con una pequeña prueba de validación, y culminando el presente documento con un resumen de posibles extensiones y mejoras.

C.2. Objetivos y requerimientos

C.2.1. Objetivos

El objetivo primordial de la herramienta, como ya se mencionó arriba, es el de servir de traductor de archivos HEIDI a GXL en ambos sentidos.
Dado además, que a simple vista este objetivo no aparenta presentar grandes dificultades más que un correcto análisis de las construcciones básicas de los dos lenguajes, nos planteamos desde un principio como un objetivo secundario, el tratar de construir una aplicación lo más simple posible. Esto trae aparejado un mayor acercamiento a otros objetivos implícitos en cualquier proyecto de desarrollo de software, como pueden ser lograr un grado importante de mantenibilidad y extensibilidad de la aplicación.

Entrando más en detalle en objetivos particulares del desarrollo, podemos mencionar los siguientes:

- Posibilidad de selección por parte del usuario de los archivos de origen y destino de la transformación de manera sencilla y amigable.
- Para la traducción de archivos GXL a archivos HEIDI, se pretende dar al usuario la posibilidad de seleccionar que atributos del archivo GXL se mapearán con los tres atributos definidos para los archivos HEIDI, tanto para los nodos como para las aristas.

C.2.2. Requerimientos

Planteados los objetivos macro de la aplicación, pasaremos a desarrollar algunos requerimientos particulares y más específicos a lo que es el diseño e implementación de la herramienta:

- Desarrollo de una única aplicación en la que se tenga la posibilidad de elegir en tiempo de ejecución que tipo de traducción (GXL-HEIDI o viceversa) se realizará.
- Selección asistida de los atributos de nodos y aristas del archivo GXL para el mapeo con los atributos fijos definidos en HEIDI, por ejemplo a través de la lectura de los mismos y su inclusión en un listado para la posterior selección por parte del usuario.
- Controles básicos de consistencia sintáctica de los archivos de entrada.
- Generación de archivos de salida según las normas sintácticas de ambos lenguajes de representación de grafos.

C.3. Marco Conceptual

C.3.1. Modelo conceptual

Como primer paso en la tarea de construcción de una herramienta que cumpla los objetivos y el alcance descritos en la sección anterior, surge la necesidad de realizar un análisis de la realidad que se pretende modelar.
Para esta tarea particular, utilizaremos las herramientas que nos brinda el modelo UML. Esta elección se realizó de forma más que natural, debido al nivel de difusión y estandarización que tienen todas las herramientas.
En particular, para el análisis y el modelado de la realidad en la que se trabajará, se decidió la utilización de modelos estáticos, específicamente, el Modelo Conceptual.

Para nuestra realidad particular utilizaremos el siguiente modelo:

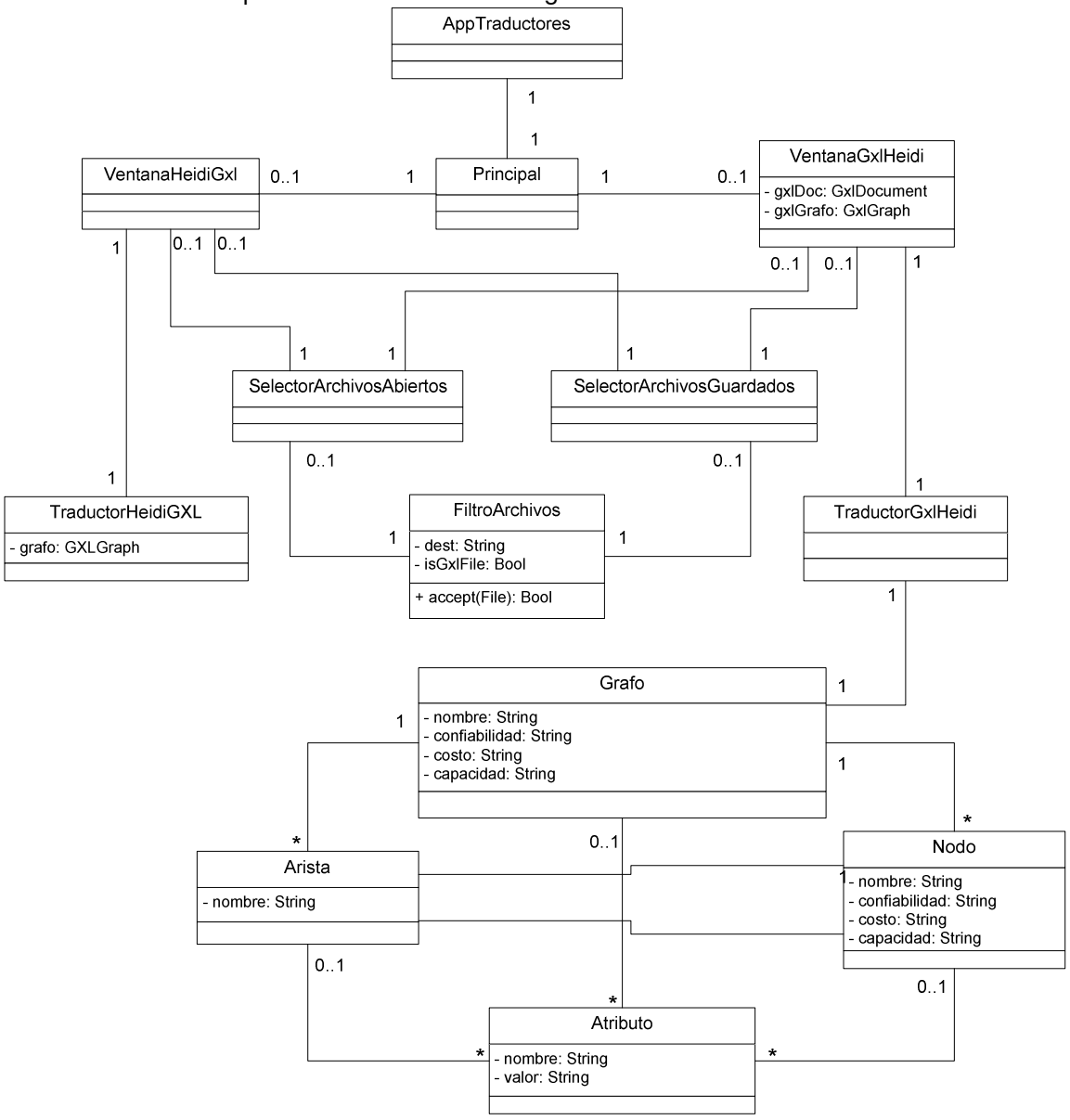


Fig. C.3: Modelo conceptual

El diagrama se puede dividir lógicamente en dos partes: una para la aplicación de traducción de archivos GXL a archivos HEIDI; y otra para la traducción inversa.

La aplicación principal crea una ventana inicial (*Principal*) en la cual se seleccionará que tipo de traducción de las anteriormente nombradas se realizará. La elección de la opción de traducción HEIDI-GXL, dará lugar a la ventana de selección de parámetros para dicho proceso (*VentanaHeidiGxl*) y a la posterior ejecución de toda la lógica de traducción, contenida en la clase *TraductorHeidiGxl*.
En el otro extremo, la elección de la opción de traducir un archivo GXL a un archivo HEIDI, está representada por la ventana de selección de parámetros para esta operación (*VentanaGxlHeidi*) y toda la estructura para la representación en memoria del grafo representado por el archivo GXL (clases *Grafo*, *Nodo*, *Arista* y *Atributo*, y las relaciones que las comunican)

Cabe mencionar la utilización de ventanas para la elección de los archivos de origen y de destino de la traducción, cumpliendo con uno de los objetivos planteados para la aplicación, las cuales se representan por las clases *SelectorArchivosAbiertos* y *SelectorArchivosGuardados* respectivamente. Se incluyó además la posibilidad de realizar un filtrado según el tipo de los archivos (extensión .gxl para los archivos GXL y extensión .exp para los archivos HEIDI), modelándose este filtro con la clase *FiltroArchivos*.

Un punto que vale la pena aclarar sobre este modelado de la realidad con que se trabajará, es la diferencia de criterios para el modelado de las realidades para cada uno de los traductores. Para el caso del traductor HEIDI-GXL se representó toda la lógica de lectura del archivo de origen traducción y escritura del archivo de destino en una única clase, mientras que para la traducción GXL-HEIDI se representó utilizando una estructura de grafo explícita para la representación del archivo GXL, a partir de la cual se realizará la traducción al archivo HEIDI.
Esta diferencia se realizó principalmente para permitir una optimización del proceso de traducción GXL-HEIDI, ya que para este caso son necesarias dos “recorridas” del archivo GXL origen: una para la obtención de los atributos del mismo (tanto a nivel del grafo, como a nivel de los nodos y aristas), y otra para la realización en sí de la traducción. Utilizando la estructura explícita de grafo que se presentó, alcanza con “recorrer” una sola vez el archivo, almacenarlo en memoria en el grafo y luego obtener la información para cada uno de los pasos de esta estructura. De esta forma se ahorrarían accesos a disco, aumentándose, por ende, la eficiencia (en tiempo de respuesta) de la aplicación.
Para la traducción HEIDI-GXL no es necesaria esta segunda “recorrida”, ya que no hay necesidad de que el usuario realice la instancia de mapeo de atributos para la traducción. Entonces podemos realizar la “recorrida” del archivo HEIDI y paralelamente ir generando el archivo GXL.

C.4. Diseño

C.4.1. Consideraciones generales de diseño

Un punto importante en cualquier desarrollo de software, es la búsqueda y estudio de herramientas que puedan a llegar a ser de utilidad en el desarrollo de la aplicación, de forma de poder adaptar el diseño de la aplicación a la utilización de dichas herramientas. En nuestro caso, el manejo de formatos de archivos fijos claramente definidos, nos llevó a la búsqueda de herramientas para el manejo de los mismos.
En el caso de los archivos HEIDI la búsqueda de herramientas no arrojó resultados de interés para nuestro caso particular.
Sin embargo para el caso de archivos GXL se cuentan con herramientas muy útiles para su procesamiento. Es así que, teniendo en mente que ya aun en este etapa de diseño ya se cuenta con un lenguaje definido para la futura implementación de la aplicación (Java), se ha decidido considerar esta herramienta para el manejo de arribos GXL, que precisamente brinda una API de Java para la lectura y creación de archivos GXL.

Reconocemos que esta toma de decisiones de diseño basadas en herramientas para lenguajes de programación particulares, no es la práctica ideal de Ingeniería de Software al momento de diseñar una aplicación. Es más, estamos conscientes de que esta decisión generará un diseño sesgado y condicionado a que luego se implemente en Java, pero nos pareció que la utilidad de la biblioteca y el hecho de que no se pretende modelar una aplicación extensible de forma automática a otros lenguajes de programación, son los justificativos de nuestra elección.
Por otro lado, un análisis superficial nos lleva a pensar que dado que GXL es un lenguaje basado en XML, sea cual sea el lenguaje de programación que se elija, es imprescindible contar por lo menos con una biblioteca para el manejo de XML. Dado que bibliotecas de este estilo se encuentran disponibles para casi cualquier lenguaje de programación difundido, la adaptación de nuestro diseño a otras herramientas, aparentemente podría realizarse sin un esfuerzo mayor de rediseño, cambiando las operaciones de la biblioteca GXL por las operaciones de la biblioteca para XML que corresponda.

Continuando con lo planteado en la sección anterior y manteniendo la consistencia con ésta, se utilizaran herramientas del modelo UML para los diagramas de diseño que se presentarán para la aplicación.
En este caso hemos decidido utilizar Diagramas de Colaboración para presentar el funcionamiento interno de las dos operaciones principales de la aplicación, o sea, la traducción de un tipo de archivos al otro.
Para complementar el Modelo Conceptual y presentar una imagen del diseño global de la aplicación, también se presentará un Diagrama de Clases de la aplicación.

En las siguientes subsecciones se presentan consecutivamente los Diagramas de Colaboración para las dos operaciones mencionadas arriba, y posteriormente el Diagrama de Clases para la aplicación.

C.4.2. Diagramas de colaboración

C.4.2.1. Traducción HEIDI-GXL

Una vez seleccionada la opción de realizar una traducción HEIDI-GXL, y habiéndose definido los archivos de origen y destino de la traducción, la ejecución del proceso de traducción se disparará por un evento externo provocado por el usuario, que un nuestro caso representaremos por el evento de presionar un botón de inicio de traducción (que en el diagrama mencionaremos como botón "*Convertir*").
Este evento obtendrá una nueva instancia de la Interfaz del traductor HEIDI-GXL (que implementa la clase *TraductorHeidiGxl*) e invocará la operación de traducción sobre la misma, pasándole los caminos absolutos de los dos archivos.

Esto se representa en el siguiente Diagrama de Colaboración:

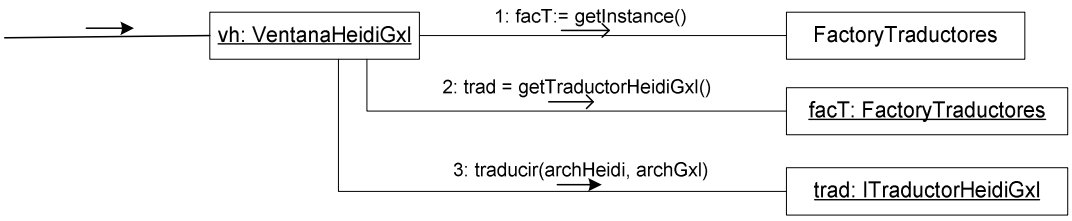


Fig. C.4: Diagrama de colaboración – VentanaHeidiGxl [Botón 'Convertir']

La lógica de la traducción, propiamente dicha, se encuentra entonces en la operación *TraductorHeidiGxl:traducir*, y se muestra a continuación:

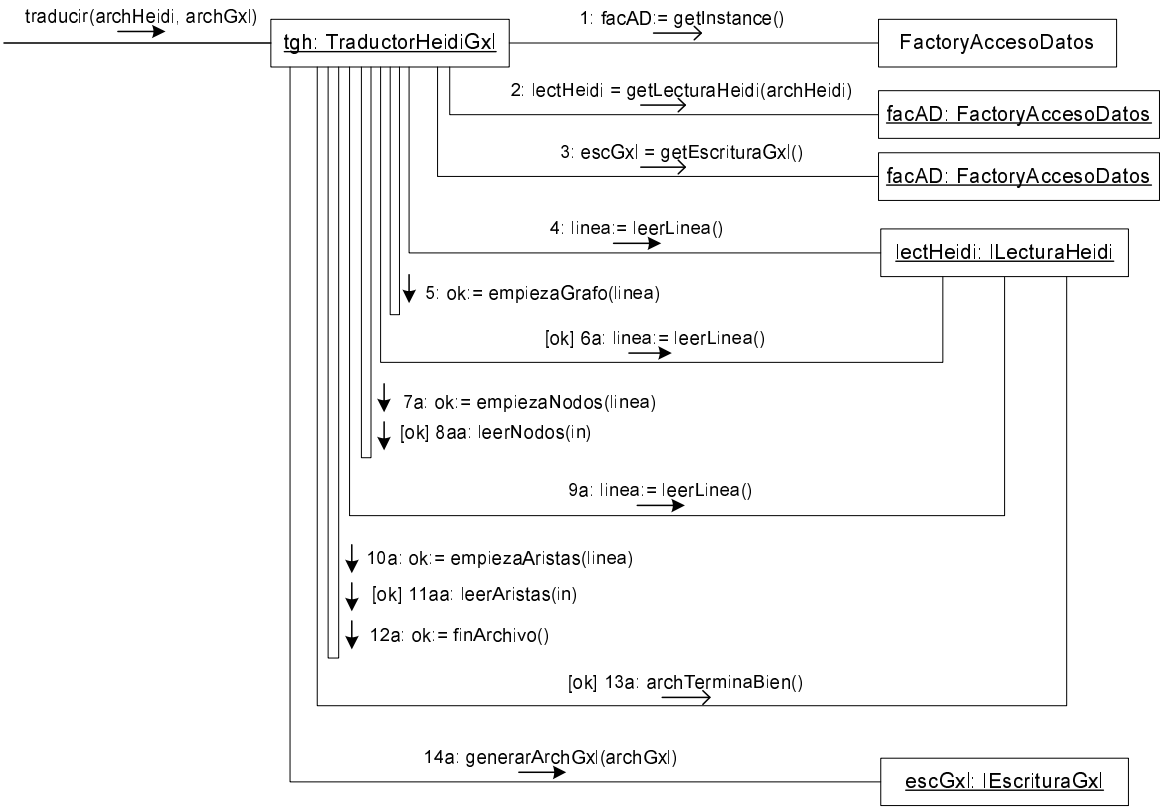


Fig. C.5: Diagrama de colaboración- TraductorHeidiGxl:treducir()

La lógica de esta traducción nos parece no amerita mayor comentario: se busca el comienzo del grafo dentro del archivo HEIDI (tag 'GRAFO') y se consume esa línea. Luego, se busca el comienzo de la declaración de nodos (tag 'CONJUNTO DE NODOS') y se procesan los mismos mediante la operación *leerNodos()*. Análogamente, se busca el comienzo de la declaración de aristas (tag 'CONJUNTO DE ARISTAS') y se procesan utilizando en este caso la operación *leerAristas()*. Por último, se chequea que todos los tags mencionados antes contengan sus correspondientes tags de cierre, para lo cual se utiliza la operación *archTerminaBien()*.
En cada una de estas operaciones, paralelamente a la lectura de los nodos y aristas del archivo HEIDI, se fue creando el grafo GXL mediante los servicios brindados por la biblioteca GXLGraph para Java [HSEW 02b]. Debido a esto, al finalizar el procesamiento del grafo, ya tenemos el grafo GXL cargado en una instancia de la clase *GXLGraph* y solo resta realizar la grabación del mismo en un archivo GXL, lo cual también se realiza utilizando operaciones brindadas por la biblioteca antes mencionada. Esta última tarea precisamente, es la que realiza la operación *generarArchGxl()* al final de la operación de traducción.
Como se habrá notado, todo el manejo tanto de la lectura del archivo HEIDI como la escritura del archivo GXL, se realizan utilizando operaciones brindadas por las interfaces *ILecturaGxl* e *IEscrituraHeidi*. La creación de estas clases tiene el objetivo de encapsular todo lo referente al acceso a los datos en una capa independiente de la aplicación, de forma de aumentar la extensibilidad de la misma.

C.4.2.2. Traducción GXL-HEIDI

Al igual que en el caso anterior, el proceso de traducción de un archivo GXL a un archivo HEIDI se dispara mediante una acción del usuario, que nuevamente representaremos en el diagrama como presionar el botón "*Convertir*".
A diferencia del caso anterior, para esta traducción es necesario que el usuario realice una elección de los atributos de los nodos y aristas del archivo GXL seleccionado que se mapearán con los atributos predefinidos para nodos y aristas en los archivos HEIDI. La elección final del usuario generará la carga de dos arreglos:

- atNodo: Contendrá tres atributos de nodos del archivo GXL que se maperán con los atributos Confiabilidad, Costo y Capacidad (en este orden) de los nodos archivo HEIDI que se genere.
- atArista: Contendrá tres atributos de aristas del archivo GXL que se maperán con los atributos Confiabilidad, Costo y Capacidad (en este orden) de las aristas del archivo HEIDI que se genere.

El comienzo del proceso de traducción se presenta en los siguientes Diagramas de Colaboración:

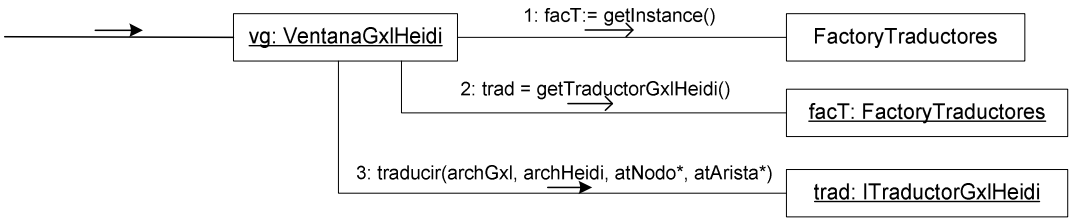


Fig. C.6: Diagrama de colaboración – VentanaGxlHeidi [Botón 'Convertir']

Algo que no comentamos al hablar de la traducción HEIDI-GXL, es el uso de las clases *FactoryTraductores*, *ITraductorGxlHeidi* e *ITraductorHeidiGxl.* Las mismas se incluyeron el diseño de la aplicación de forma de lograr un nivel de independencia de capas adecuado, en este caso, entre las capas de presentación y lógica.

El uso de *FactoryTraductores* nos permite tener un único punto de acceso a la capa lógica desde la capa de presentación, mientras que las interfaces de los traductores nos permiten invocar las operaciones de traducción independientemente de la implementación subyacente que tengan. Esto último facilitaría la tarea de cambiar la lógica de traducción, ya que estos cambios no requerirían cambio alguno a nivel de la presentación, siempre y cuando se mantenga la interfaz.

A continuación presentaremos lo que sería el procedimiento de traducción GXL-HEIDI:

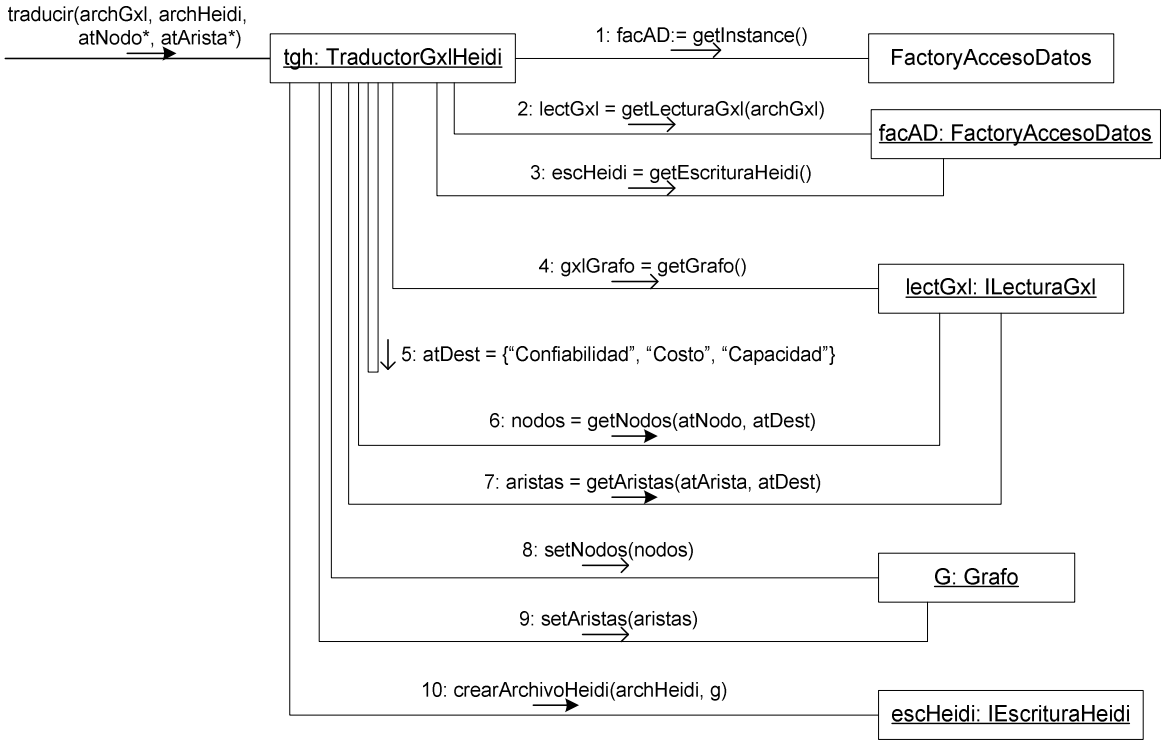


Fig. C.7: Diagrama de colaboración – TraductorGxlHeidi:traducir()

C.4.3. Diagrama de clases de diseño

Para cerrar lo que es el capítulo de diseño de la aplicación, presentaremos el Diagrama de clases de diseño de la misma.

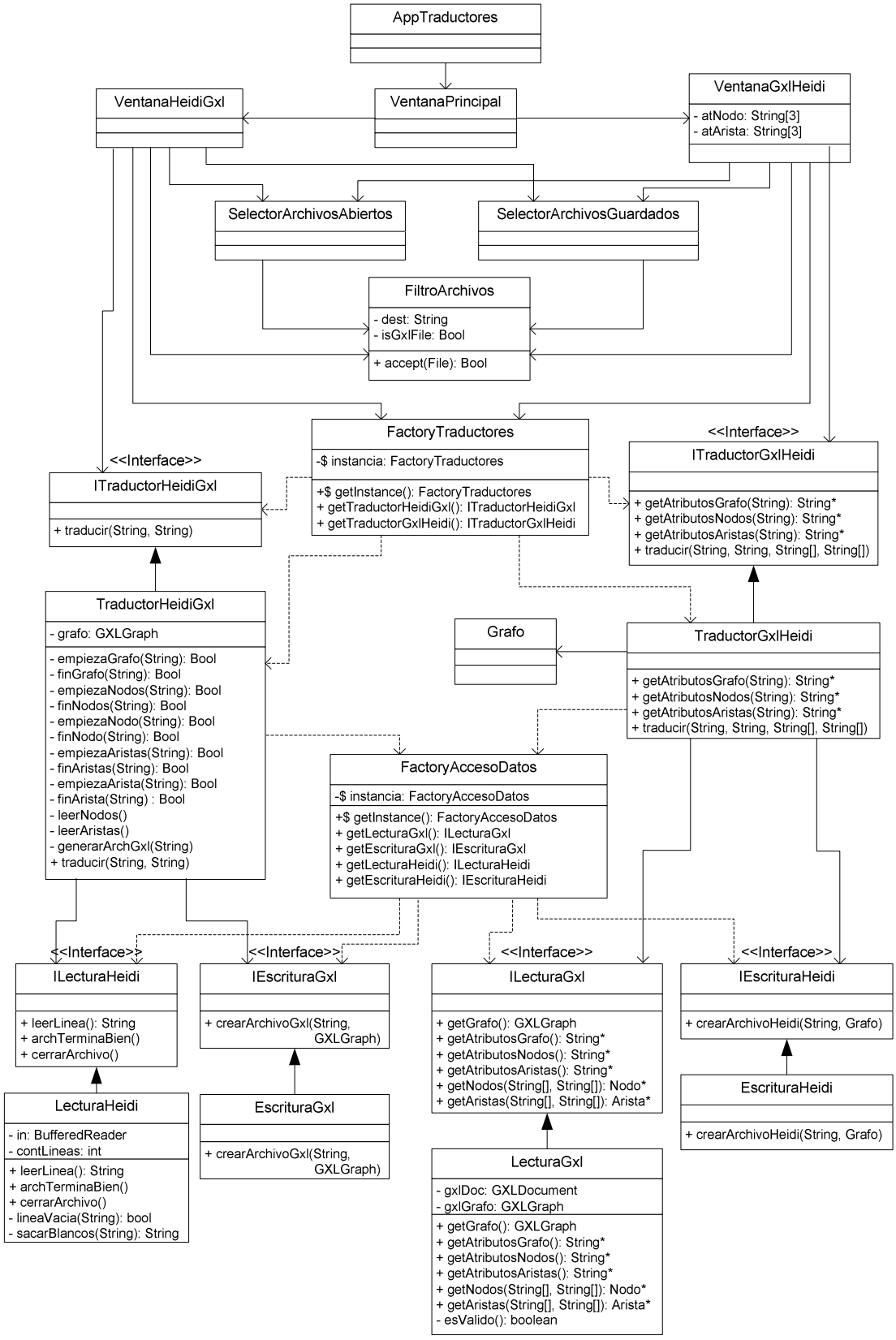


Fig. C.8a: Diagrama de clases de diseño

Separamos del anterior diagrama (por motivos de espacio), la estructura que data types que utilizamos para almacenar el archivo GXL leído durante el proceso de traducción GXL-HEIDI. La misma tiene el siguiente diseño:

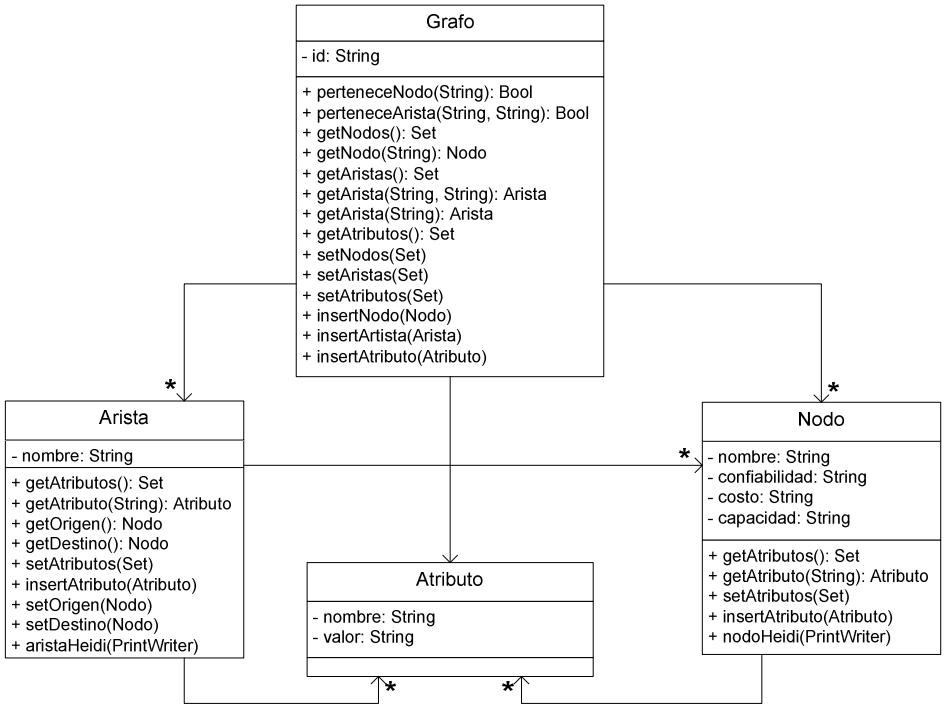


Fig. C.9b: Diagrama de clases de diseño

C.5. Prueba de Validación

C.5.1. Traductor Heidi-GXL

Para la prueba de traducción Heidi-Gxl se crean dos archivos Heidi, uno de nombre '*Puente.exp*' y otro '*Arpanet.exp*'. El primero es un grafo en el que todos los nodos y aristas tienen un valor para la confiabilidad, costo y capacidad. El segundo es un archivo que representa un grafo, como los que trabajamos en el resto del proyecto, es decir las aristas tienen un único atributo que es la confiabilidad, y los nodos no tienen atributos.

Del archivo '*Puente.exp*' se crearon varias copias de nombres: '*Puente-TagArista.exp*', '*Puente-TagFinNodo.exp*', '*Puente-FinConjunto.exp*', '*Puente-TagGrafo.exp*', y '*Puente-CostoNodo.exp*', los cuales son todos inválidos desde el punto de vista sintáctico, dado que tienen un error en la parte referenciada por su respectivo nombre.

Todos los archivos aquí mencionados se encuentran disponibles tanto en el CD del proyecto como en la página web.

Caso 1

El archivo '*Puente.exp*' es el siguiente:

```
GRAFO
CONJUNTO DE NODOS
NODO
1
0.5
20
150
FINNODO
NODO
2
0.9
25
160
FINNODO
NODO
3
0.7
10
170
FINNODO
NODO
4
0.3
15
140
FINNODO
FINCONJUNTO
CONJUNTO DE ARISTAS
ARISTA
1
2
a
0.8
55
200
FINARISTA
ARISTA
1
3
b
0.8
45
205
FINARISTA
ARISTA
2
3
c
0.6
65
210
FINARISTA
ARISTA
2
4
d
0.9
75
215
FINARISTA
ARISTA
3
4
e
0.9
55
220
FINARISTA
FINCONJUNTO
FINGRAFO
```

Fig. C.10: Grafo Puente en formato Heidi

Y el archivo puente.gxl generado por GXL Translator, fue el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
  <graph id="Grafo generado por la aplicación de traduccion HEIDI-GXL">
    <node id="1">
      <attr name="confiabilidad">
            <float>0.5</float>
      </attr>
      <attr name="costo">
            <float>20.0</float>
      </attr>
      <attr name="capacidad">
            <float>150.0</float>
      </attr>
    </node>
    <node id="2">
      <attr name="confiabilidad">
            <float>0.9</float>
      </attr>
      <attr name="costo">
            <float>25.0</float>
      </attr>
      <attr name="capacidad">
            <float>160.0</float>
      </attr>
    </node>
    <node id="3">
      <attr name="confiabilidad">
            <float>0.7</float>
      </attr>
      <attr name="costo">
            <float>10.0</float>
      </attr>
      <attr name="capacidad">
            <float>170.0</float>
      </attr>
    </node>
    <node id="4">
      <attr name="confiabilidad">
            <float>0.3</float>
      </attr>
      <attr name="costo">
            <float>15.0</float>
      </attr>
      <attr name="capacidad">
            <float>140.0</float>
      </attr>
    </node>
    <edge id="a" to="2" from="1">
      <attr name="confiabilidad">
            <float>0.8</float>
      </attr>
      <attr name="costo">
            <float>55.0</float>
      </attr>
      <attr name="capacidad">
            <float>200.0</float>
      </attr>
    </edge>
    <edge id="b" to="3" from="1">
      <attr name="confiabilidad">
            <float>0.8</float>
      </attr>
      <attr name="costo">
            <float>45.0</float>
      </attr>
      <attr name="capacidad">
            <float>205.0</float>
      </attr>
    </edge>
    <edge id="c" to="3" from="2">
      <attr name="confiabilidad">
            <float>0.6</float>
      </attr>
      <attr name="costo">
            <float>65.0</float>
      </attr>
      <attr name="capacidad">
            <float>210.0</float>
      </attr>
    </edge>
    <edge id="d" to="4" from="2">
      <attr name="confiabilidad">
            <float>0.9</float>
      </attr>
      <attr name="costo">
            <float>75.0</float>
      </attr>
      <attr name="capacidad">
            <float>215.0</float>
      </attr>
    </edge>
    <edge id="e" to="4" from="3">
      <attr name="confiabilidad">
            <float>0.9</float>
      </attr>
      <attr name="costo">
            <float>55.0</float>
      </attr>
      <attr name="capacidad">
            <float>220.0</float>
      </attr>
    </edge>
  </graph>
</gxl>
```

Fig. C.11: Grafo Puente convertido a GXL

El cual fue abierto con el visualizador, para verificar más fácilmente que sea válido y que este correctamente convertido obteniéndose la siguiente visualización:

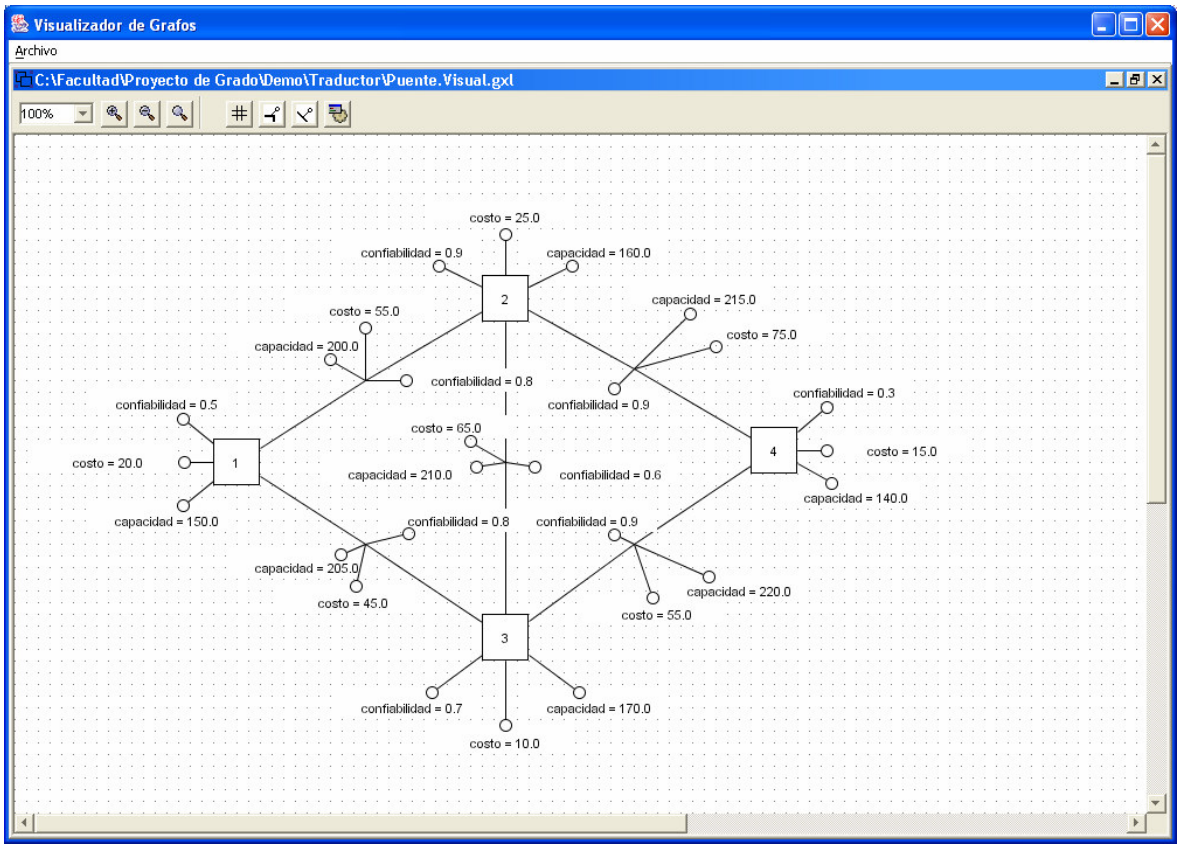


Fig. C.12: Grafo Puente visualizado con GXL Viewer (archivo: Puente.visual.gxl)

Como se puede verificar la conversión fue correcta.

Caso 2

La idea de esta prueba es tomar un grafo en formato Heidi en el cual sus nodos y aristas no tengan todos los atributos de confiabilidad, costo y capacidad. Se eligió entonces el archivo '*Arpanet.exp*' (representa la topología de dicha red en1972), el cual además de tener esa propiedad fue usado en la validación de CDA Evaluator. Este archivo se presenta en la Figura C.13.

En él los nodos no tienen atributos y las aristas tienen como único atributo la confiabilidad. El resultado es el archivo '*Arpanet.gxl*' (Figura C.14), el cual se verificó que se genero correctamente. Inclusive se visualizó con GXL Viewer y se evaluó con la funcionalidad de evaluación integrada al evaluador.

GRAFO CONJUNTO DE NODOS NODO 0 FINNODO NODO 1 FINNODO NODO 2	FINNODO NODO 15 FINNODO NODO 16 FINNODO NODO 17	ARISTA 4 5 f 0.99 FINARISTA ARISTA 5 6 g 0.99 FINARISTA ARISTA 4	ARISTA 12 16 q 0.99 FINARISTA ARISTA 13 15 r 0.99 FINARISTA ARISTA 15

FINNODO NODO 3	FINNODO NODO 18	8 h 0.99	16 s 0.99
FINNODO NODO 4	FINNODO NODO 19	FINARISTA ARISTA 6 7 i 0.99	FINARISTA ARISTA 14 20 t 0.99
FINNODO NODO 5	FINNODO NODO 20	FINARISTA ARISTA 7 10 j 0.99	FINARISTA ARISTA 20 19 u 0.99
FINNODO NODO 6	FINNODO FINCONJUNTO CONJUNTO DE ARISTAS ARISTA 0 1 a 0.99	FINARISTA ARISTA 8 11 k 0.99	FINARISTA ARISTA 19 18 v 0.99
FINNODO NODO 7	FINARISTA ARISTA 0 2 b 0.99	FINARISTA ARISTA 10 11 l 0.99	FINARISTA ARISTA 18 17 w 0.99
FINNODO NODO 8	FINARISTA ARISTA 1 3 c 0.99	FINARISTA ARISTA 10 13 m 0.99	FINARISTA ARISTA 17 16 x 0.99
FINNODO NODO 9	FINARISTA ARISTA 1 6 d 0.99	FINARISTA ARISTA 11 14 n 0.99	FINARISTA ARISTA 1 2 y 0.99
FINNODO NODO 10	FINARISTA ARISTA 2 4 e 0.99	FINARISTA ARISTA 6 9 o 0.99	FINARISTA ARISTA 3 4 z 0.99
FINNODO NODO 11	FINARISTA	FINARISTA ARISTA 9 12 p 0.99	FINARISTA FINCONJUNTO FINGRAFO
FINNODO NODO 12		FINARISTA	
FINNODO NODO 13			
FINNODO NODO 14			

Fig. C.13: Grafo Arpanet en formato Heidi

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl>
	<graph id="Grafo generado por la aplicacion de traduccion HEIDI-GXL">
		<node id="0"/>
		<node id="1"/>
		<node id="2"/>
		<node id="3"/>
		<node id="4"/>
		<node id="5"/>
		<node id="6"/>
		<node id="7"/>
		<node id="8"/>
		<node id="9"/>
		<node id="10"/>
		<node id="11"/>
		<node id="12"/>
		<node id="13"/>
		<node id="14"/>
		<node id="15"/>
		<node id="16"/>
		<node id="17"/>
		<node id="18"/>
		<node id="19"/>
		<node id="20"/>
		<edge id="a" to="1" from="0">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="b" to="2" from="0">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="c" to="3" from="1">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="d" to="6" from="1">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="e" to="4" from="2">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="f" to="5" from="4">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="g" to="6" from="5">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="h" to="8" from="4">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="i" to="7" from="6">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="j" to="10" from="7">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="k" to="11" from="8">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="l" to="11" from="10">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="m" to="13" from="10">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="n" to="14" from="11">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="o" to="9" from="6">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="p" to="12" from="9">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="q" to="16" from="12">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="r" to="15" from="13">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="s" to="16" from="15">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="t" to="20" from="14">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="u" to="19" from="20">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="v" to="18" from="19">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="w" to="17" from="18">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="x" to="16" from="17">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="y" to="2" from="1">
			<attr name="confiabilidad">
				<float>0.99</float>
			</attr>
		</edge>
		<edge id="z" to="4" from="3">
			<attr name="confiabilidad">
				<float>0.99</float>
```

	</attr> </edge> </graph> </gxl>

Fig. C.14: Grafo Arpanet traducido a GXL

Caso 3

Para los archivos cuya sintaxis es incorrecta se obtuvieron los siguientes mensajes de error:

a) '*Puente-TagFinNodo.exp*': El nodo "2" dice "FIN" donde debería decir "FINNODO".

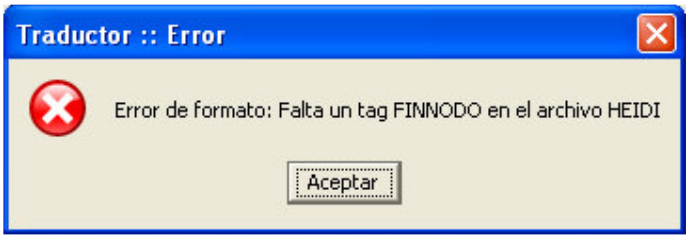


Fig. C.15: Error de formato de archivo Heidi en tag FinNodo

b) '*Puente-TagArista.exp*': La arista "b" dice "ARITA" donde debería decir "ARISTA"

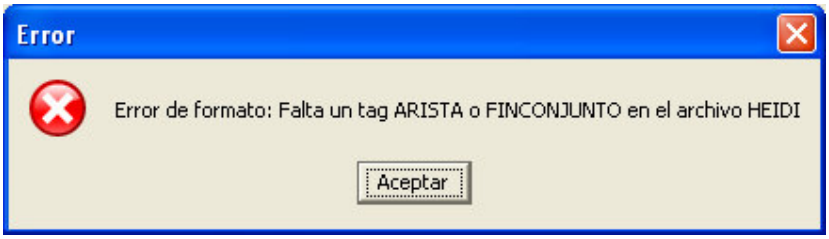


Fig. C.16: Error de formato de archivo Heidi en tag Arista

c) '*Puente-TagFinConjunto*': Falta el tag de FINCONJUNTO

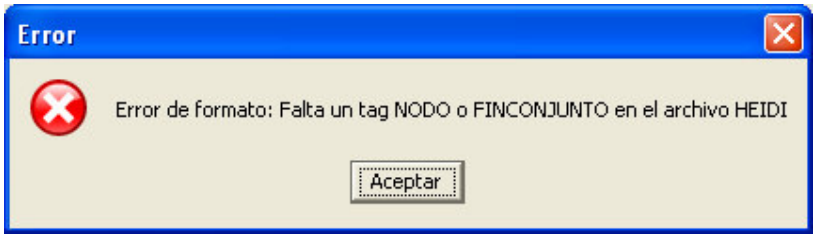


Fig. C.17: Error de formato de archivo Heidi en tag FinConjunto

d) '*Puente-CostoNodo*': Falta la línea correspondiente al costo del nodo 2.

Fig. C.18: Error de formato de archivo Heidi en el costo del nodo

e) '*Puente-TagGrafo*': El grafo comienza con el string "GRAPH" en vez de "GRAFO".

Fig. C.19: Error de formato de archivo Heidi en tag Grafo

C.5.2. Traductor GXL-Heidi

En este sentido de la traducción documentaremos 4 casos de prueba, el primero de ellos es la traducción del archivo '*Puente.gxl*' en el sentido inverso, el segundo es la conversión un grafo GXL genérico llamado '*RedTransporte.gxl*', el tercero es intentar abrir un gxl invalido de nombre '*PuenteNoValido.gxl*' y finalmente el último caso de prueba fue intentar generar un archivo Heidi en una carpeta de solo lectura.

Caso 1

Se tomó el archivo '*Puente.gxl*', el cual fue generado a partir de '*Puente.exp*' por el traductor Heidi-GXL, y se generó un archivo de nombre '*PuenteConvertido.exp*', para verificar que ambos archivos son iguales.

El archivo Heidi generado fue el de la Figura C.10, que como se observa es igual al original '*Puente.gxl*', a excepción del orden en el que aparecen los nodos.

GRAFO	ARISTA
CONJUNTO DE NODOS	1
NODO	3
3	b
0.7	0.8
10.0	45.0
170.0	205.0
FINNODO	FINARISTA
NODO	ARISTA
4	2
0.3	4
15.0	d
140.0	0.9
FINNODO	75.0
NODO	215.0
2	FINARISTA
0.9	ARISTA
25.0	2

160.0 FINNODO NODO 1 0.5 20.0 150.0 FINNODO FINCONJUNTO CONJUNTO DE ARISTAS ARISTA 3 4 e 0.9 55.0 220.0 FINARISTA	3 c 0.6 65.0 210.0 FINARISTA ARISTA 1 2 a 0.8 55.0 200.0 FINARISTA FINCONJUNTO FINGRAFO

Fig. C.20: PuenteConvertido.exp es igual al Puente.exp original

Caso 2

La conversión del grafo '*RedTransporte.gxl*' representa un caso genérico de conversión donde:

- Al hacer la unión de los todos atributos de nodo o las arista, quedan más de 3 atributos, es decir hay atributos que no se van a mapear al archivo Heidi. En particular para las aristas quedan "confiabilidad", "peaje", "diámetro" y "perímetro".

- No todos los nodos o aristas tienen todos los atributos, por lo tanto al elegirse un atributo no presente en todos los nodos o aristas debería imprimirse una línea en blanco en el archivo Heidi. En este caso se dará este hecho con el atributo diámetro de las aristas.

- Se tienen atributos a nivel de grafo que, al elegirse para el mapeo, debería aparecer en todos los nodos o aristas y con el mismo valor. En este caso se toma el atributo capacidad del grafo y se lo mapeo a la capacidad de los nodos.

- Se decide no mapear ningún atributo del archivo gxl a un determinado atributo del archivo Heidi. El atributo costo de nodo no es mapeado a ningún atributo del GXL.

- Las aristas no tienen "id" y por tanto se genera una línea en blanco en su lugar en el archivo Heidi (el grafo del caso 1 prueba el caso en que si tienen identificador).

- Los atributos en el GXL son de diferente tipo. En particular los atributos "pedidos" y "peaje" son tipo "int" y los demás son tipo "float"

El archivo '*RedTransporte.gxl*' se presenta en la Figura C.21 y el mapeo seleccionado es el de la figura C.20

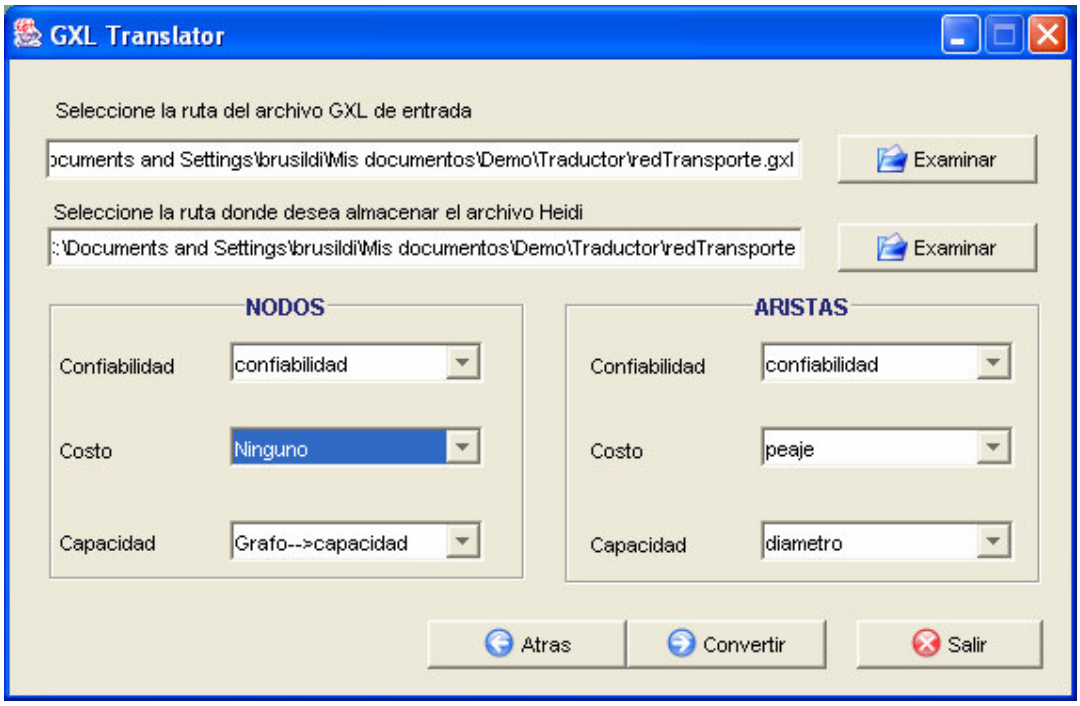


Fig. C.21: Conversión del grafo RedTransporte de Heidi a GXL

```
<?xml version="1.0" encoding="UTF-8"?>
<gxl>
        <graph id="identificador del grafo">
                <attr name="capacidad">
                        <float>90</float>
                </attr>
                <node id="1">
                        <attr name="confiabilidad">
                                <float>0.5</float>
                        </attr>
                        <attr name="carga">
                                <float>10</float>
                        </attr>
                        <attr name="pedidos">
                                <int>100</int>
                        </attr>
                </node>
                <node id="2">
                        <attr name="confiabilidad">
                                <float>0.6</float>
                        </attr>
                        <attr name="pedidos">
                                <int>120</int>
                        </attr>
                </node>
                <node id="3">
                        <attr name="confiabilidad">
                                <float>0.9</float>
                        </attr>
                        <attr name="carga">
                                <float>20</float>
                        </attr>
                        <attr name="pedidos">
                                <int>200</int>
                        </attr>
                </node>
                <node id="4">
                        <attr name="confiabilidad">
                                <float>0.8</float>
                        </attr>
                        <attr name="pedidos">
                                <int>125</int>
                        </attr>
                </node>
                <edge from="1" to="2">
                        <attr name="confiabilidad">
                                <float>0.05</float>
                        </attr>
                        <attr name="diametro">
                                <float>5</float>
                        </attr>
                        <attr name="peaje">
                                <int>20</int>
                        </attr>
                </edge>
                <edge from="1" to="3">
                        <attr name="confiabilidad">
                                <float>0.4</float>
                        </attr>
                        <attr name="perimetro">
                                <float>7</float>
                        </attr>
                        <attr name="peaje">
                                <int>20</int>
                        </attr>
                </edge>
                <edge from="1" to="4">
                        <attr name="confiabilidad">
                                <float>0.4</float>
                        </attr>
                        <attr name="peaje">
                                <int>25</int>
                        </attr>
                        <attr name="diametro">
                                <float>8</float>
                        </attr>
                </edge>
                <edge from="2" to="4">
                        <attr name="confiabilidad">
                                <float>0.6</float>
                        </attr>
                        <attr name="peaje">
                                <int>30</int>
                        </attr>
                        <attr name="diametro">
                                <float>8</float>
                        </attr>
                </edge>
        </graph>
</gxl>
```

Fig. C.22: RedTransporte en formato GXL

El resultado de la conversión se muestra en la Figura C.21, donde se puede apreciar que el costo de los nodos esta en blanco, la confiabilidad global del grafo es la de todos los nodos, y cada confiabilidad de nodo y arista fue mapeada a la respectiva del archivo Heidi. El diámetro de las aristas fue mapeado a la capacidad y la arista <1,3> al no tener diámetro muestra una línea en blanco. Los atributos carga y pedidos de nodos y perímetro de aristas, efectivamente no fueron tenidos en cuenta.

GRAFO CONJUNTO DE NODOS NODO 4 0.8 90 FINNODO NODO 1 0.5 90 FINNODO NODO 2 0.6 90 FINNODO NODO 3 0.9 90 FINNODO FINCONJUNTO CONJUNTO DE ARISTAS	ARISTA 1 2 0.05 20 5 FINARISTA ARISTA 2 4 0.6 30 8 FINARISTA ARISTA 1 3 0.4 20 FINARISTA ARISTA 1 4 0.4 25 8 FINARISTA FINCONJUNTO FINGRAFO

Fig. C.23: RedTransporte.exp en formato Heidi

Caso 3

a) Se intento convertir el archivo '*RedTransporteNoValida.gxl*', donde hay un tag </edge> donde debería decir </node>, y la aplicación desplegó el mensaje de la Figura C.24 al intentar leer el archivo para cargar los combos.

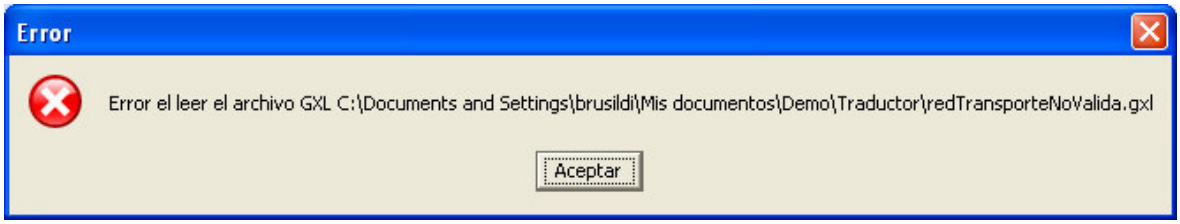


Fig. C.24: Error al intentar abrir un archivo GXL no válido

Aclaración: Si el grafo GXL es válido, pero no es "simple" por ejemplo tiene aristas n-arias o es un multigrado, es decir hay nodos con grafos en él, la aplicación no captura la situación y el resultado es indefinido. La funcionalidad de validar que sea un grafo "simple" mediante un DTD es posible pero se dejo marcado para un trabajo futuro.

b) También se probó de intentar crear un archivo Heidi en una carpeta donde no se tiene derechos de escritura y se obtuvo el mensaje de error de la Figura C.25

Fig. C.25: Error al intentar crear el archivo Heidi

A su vez se intento validar que todas las secuencias de eventos posibles sean bien manejadas por la aplicación como por ejemplo, que al intentar hacer una conversión sin seleccionar un grafo GXL o un archivo Heidi, la aplicación capture e informe la situación, que las traducciones posteriores a la primera también funcionen bien (no sean afectadas por la traducción anterior) y que al cancelar un diálogo de Abrir o Guardar archivo el resultado sea el esperado.

C.5. Conclusiones y trabajo futuro

La aplicación de traducción entre archivos GXL y HEIDI pretende ser una herramienta sencilla y de fácil uso, tomando como base que el problema que intenta resolver, en sí mismo no presenta mayores complicaciones.
Dentro de este marco de trabajo es común perder de vista aspectos básicos para el desarrollo de aplicaciones que con un cierto nivel de calidad, obteniéndose aplicaciones monolíticas que posteriormente presentan problemas de mantenibilidad y extensibilidad. Teniendo en mente que la aplicación seguramente pueda ser mantenida por otras personas ya que forma parte de un proyecto macro, y tratando de no caer en la tendencia de desarrollar una aplicación en bloque, se intentó hacer una separación en las tres capas clásicas: Presentación, Lógica y Acceso a datos.

El aspecto más importante que consideramos para una extensión de la aplicación es la utilización de algún mecanismo más sofisticado de validación de los archivos de origen de la traducción; por ej, utilizar un archivo DTD o un XML Schema para validar el archivo GXL.

C.6. Referencias

[CU 93]
"HEIDI – Una herramienta de apoyo para el diseño de redes de comunicación"
H. Cancela, M. Urquhart.
Technical report INCO 93-01, 1993.

[HSEW 02]
Graph Exchange Lenguaje, Background.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/Introduction/Backgound.html, Julio de 2002.

[HSEW 02b]
Graph Exchange Lenguaje, Tool Catalogue.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL/tools/tools.html, Julio de 2002.

[Rodríguez 02]
"Lenguajes canónicos para la descripción de grafos: estudio y transformación entre esquemas de distintos modelos de datos"
Pablo Rodríguez.
InCo, Setiembre de 2002.

[WWW 98]
"Extensible Markup Language (XML) 1.0",
World Wide Web Consortium. Editores T. Bray, J. Paoli, C.M. Sperberg, McQueen
http://www.w3.org/TR/REC-xml, Febrero de 1998.

:: Apéndice D

:: Visualizador de Grafos: GXL Viewer

D.1. Introducción

El presente documento es una descripción a grandes rasgos de la arquitectura y el diseño de la aplicación para la visualización de grafos, que denominaremos a lo largo de este informe como 'VisualizadorGrafos'.
En las siguientes secciones desarrollaremos los objetivos y alcances de la aplicación, la arquitectura de la misma, y por último haremos una síntesis de su diseño y algunos comentarios sobre la implementación y el funcionamiento de sus clases más importantes.

Cabe aclarar que para la implementación de la aplicación se tomó como base los subsistemas de visualización de grafos de los sistemas *'SIM Engine'* [PIS 02] y *'EDM5'* [PIS 03] desarrollados respectivamente durante los Proyectos de Ingeniería de Software de los años 2002 y 2003. En ambos casos se utilizó la misma base para la representación de estructuras de grafo tanto en memoria como de forma gráfica, las cuales se basan en la utilización de las bibliotecas externas JGraph [JGp 00] y SDA-Graph [SDA 00].
El procedimiento básico para la construcción de la aplicación fue tomar el subsistema de visualización de grafos del '*EDM5*', independizarlo del sistema y posteriormente adaptarlo a las necesidades particulares de nuestro desarrollo (lectura de archivos GXL, adaptación de las estructuras que se representarán en los grafos que se manejan, adaptación de la interfaz gráfica a los requerimientos del presente proyecto, etc.)

El presente documento no pretende ser una reingeniería de software que a partir del código reconstruya el diseño de la aplicación para su posterior adaptación, sino que solamente pretende hacer las veces de documento de diseño de la aplicación que finalmente se construyó de forma de facilitar futuros trabajos sobre la misma.

D.2. Objetivos y requerimientos

D.2.1. Objetivos

La herramienta GXL Viewer tiene básicamente dos objetivos en su desarrollo: leer archivos GXL y representar los mismos en una estructura de memoria determinada, y luego desplegar gráficamente el grafo representado en el archivo GXL leído.
Si miramos el desarrollo de esta herramienta en el marco del proyecto que forma parte, resulta de interés la posibilidad de aplicar los diferentes algoritmos de evaluación que se desarrollaron (el algoritmo de optimización no es aplicable en este contexto, ya que toma como entrada la descripción genérica de una familia de grafos, mientras que el Visualizador está pensado para la representación de instancias de grafos particulares) para lo cual se buscará la integración con los algoritmos implementados para el Evaluador. Esto servirá también, de alguna forma, para probar la calidad en cuanto a la extensibilidad del Evaluador.

Además de estos objetivos particulares de la aplicación, se intentaron seguir los objetivos básicos para el desarrollo de software de cualquier aplicación, como el mantenimiento de la independencia de capas, la mantenibilidad y extensibilidad de las clases, y la eliminación de la mayor cantidad de bugs posibles.

Dentro de los dos grandes objetivos planteados para la aplicación, se podrían mencionar otros objetivos secundarios:

- Posibilidad de elección del archivo de forma amigable para el usuario.
- Brindar algún algoritmo de ordenación para la representación gráfica del grafo.
- Desarrollo de una interfaz gráfica que permita la manipulación básica del grafo.
- Permitir atributos en nodos y aristas con la posibilidad de visualizar los mismos de forma opcional.

D.2.2. Requerimientos

Entrando en lo que son los requerimientos para la aplicación, podemos mencionar la siguiente lista:

- Visualización gráfica de grafos cuya información se obtiene de archivos GXL.
- Manipulación amigable de la representación gráfica del grafo.
- Ordenamiento básico de los elementos gráficos utilizados para la representación de nodos, aristas y atributos del grafo, dando igualmente al usuario la posibilidad de disponer de los mismos en la ventana a su gusto y de guardar una cierta disposición.
- Posibilitar la aplicación de los algoritmos de evaluación implementados sobre el grafo que se esté visualizando.
- Visualización opcional de los atributos de nodos y aristas.

D.3. Diseño

D.3.1. Consideraciones generales de diseño

A pesar de contarse con la posibilidad de utilizar varias bibliotecas auxiliares (representación de grafos en memoria, representación gráfica de grafos y manejo de archivos GXL) la relativa complejidad y el importante tamaño de la aplicación nos llevaron a realizar una organización de los diagramas y los comentarios que correspondan en cada caso, primero por capa e internamente dentro de cada capa por subsistema o paquete.

Para el análisis que realizaremos a lo largo de toda la sección de Diseño, se harán algunas flexibilizaciones con respecto a lo que sería un documento de diseño “clásico”.
Por un lado la división de subsistemas de la aplicación coincide con los paquetes que se implementaron, por lo que, además de mantener la nomenclatura de los paquetes para referirnos a los subsistemas, se emplearán los términos ‘subsistema’ y ‘paquete’ de forma indistinta.
Además durante la descripción de los diferentes subsistemas, se adjuntarán a los comentarios referentes a las decisiones de diseño que se tomaron, algunos referentes a decisiones de implementación de las clases involucradas.

Comenzaremos desarrollando el diseño de la capa de Presentación (o Gráfica), la cual por su tamaño y complejidad es la más importante del sistema. Continuaremos, siguiendo el esquema de capas, por la capa Lógica, para finalizar con la capa de Acceso de Datos.

Como una introducción a lo que es la arquitectura del sistema, a continuación en la Figura D.1 presentamos una vista del modelo de diseño del sistema con los subsistemas que identificamos en el mismo y las dependencias entre los mismos:

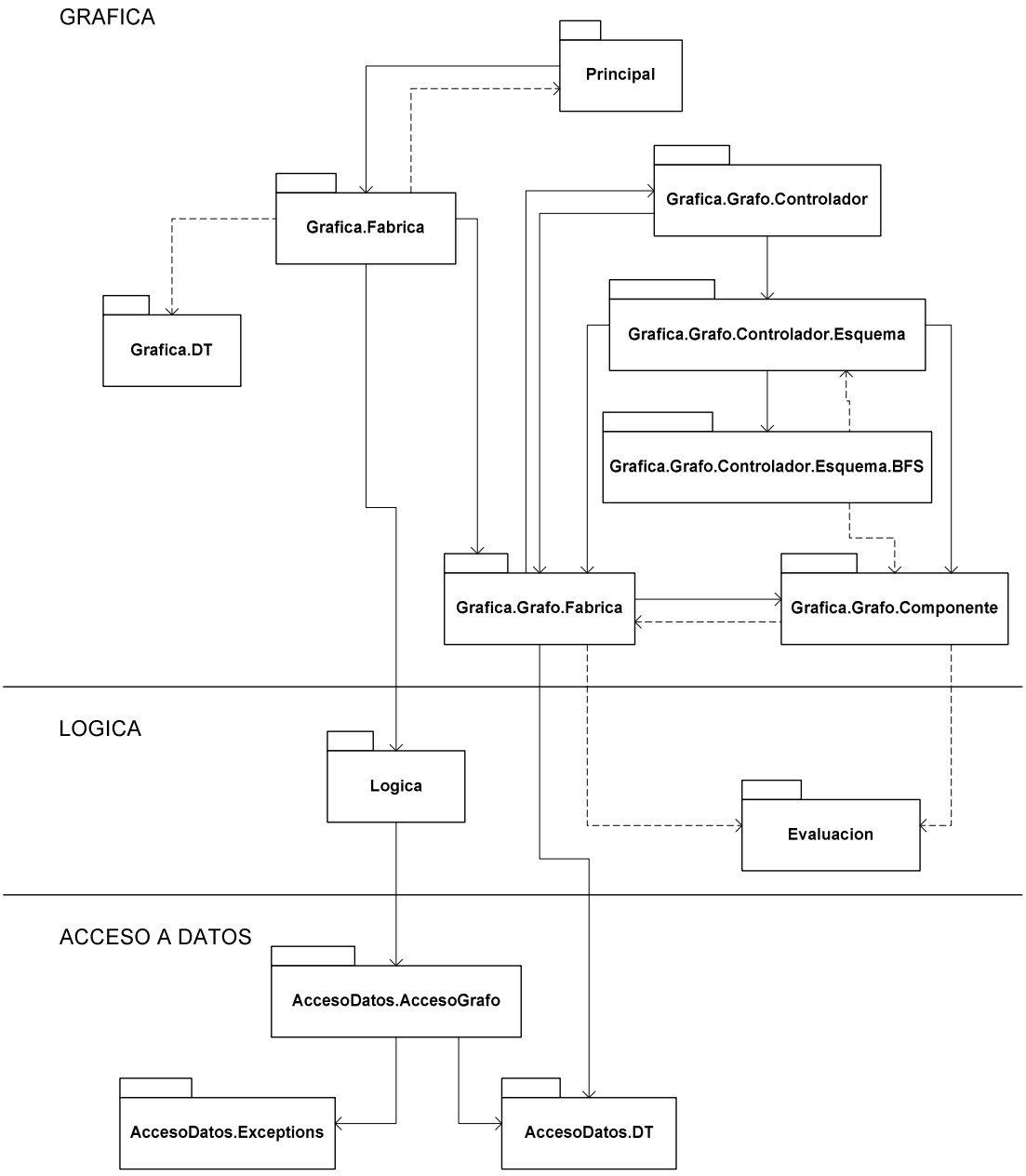


Fig. D.1: Vista del modelo de diseño

Como es intuitivo teniendo en vista la estructura de paquetes, prácticamente todo el desarrollo del sistema está concentrado en la capa Gráfica, en donde se realiza todo el trabajo de representación en memoria del grafo a visualizar y de representación gráfica del mismo (incluyendo la ordenación de los elementos gráficos de forma relativamente armoniosa en la ventana).
La capa Lógica prácticamente no realiza trabajo alguno, sirviendo como puente con la capa de Acceso a Datos en la cual se realiza la lectura y escritura de los archivos GXL que se utilizan como fuente de almacenamiento de la información de los diferentes grafos. En la capa Lógica también se incluye todo el subsistema para evaluación de grafos que se fusionó con el Visualizador para permitir la evaluación sobre los grafos que se visualicen.

En las siguientes sub-secciones se entrará en mayores detalles del funcionamiento interno de cada capa.
Para cada una de las capas y subsistemas que se identificaron se presentará un diagrama de clases de la misma. Hemos tomado la convención de presentar las clases que corresponden a la capa o subsistema de color negro, mientras que las clases relacionadas directamente con ellas pero no pertenecientes al sistema se muestran en azul. Esta es una referencia que seguiremos a lo largo de todo este documento.

D.3.2. Capa de Presentación

Básicamente la capa de presentación se encarga de todo lo necesario para la representación de la información referente a un grafo que se obtuvo desde un archivo GXL de forma visual en la ventana del programa. Para ello es necesario recorrer un camino, desde la obtención de la información del grafo a través de algún Data Type adecuado, hasta la representación grafica en sí misma de esa información (formas asociadas a cada objeto, posición de cada objeto, etc.).

Para un mejor orden en el desarrollo dividiremos la sección según los diferentes subsistemas que podemos identificar en la capa (utilizamos los nombres de los paquetes para identificar cada uno de los subsistemas que definimos):

- Principal: Contiene el main de la aplicación, que corresponde con la ventana principal de la misma, además de algunas clases accesorias para el mismo, como la barra de menú que se asoció a la misma, las ventanas internas que se utilizarán para desplegar los grafos y las ventanas para selección de los archivos que se visualizan. En este subsistema no se realiza ningún trabajo de lógica más del necesario para configurar las ventanas.
- Grafica.Fabrica: Define una interfaz para el acceso a las operaciones básicas del programa: abrir, guardar y cerrar un grafo determinado. Además se brinda una implementación particular para estas operaciones.
- Grafica.Grafo.Fabrica: Define un conjunto de interfaces para el acceso a las operaciones particulares sobre un cierto grafo: agregado de nodos y aristas, posicionamiento de los mismos, selección y deselección de los objetos, entre otras.
- Grafica.Grafo.Controlador: Implementación particular de la interfaz de acceso a las operaciones del grafo que acabamos de mencionar. En este subsistema se incluye el posicionamiento de los elementos del grafo, su ordenamiento en la pantalla, el manejo de las selecciones y el movimiento de los objetos del grafo. El objetivo de toda la lógica de este subsistema es crear una representación del grafo en memoria con toda esta información (lo que hemos llamado esquema del grafo) utilizando las clases brindadas por la biblioteca SDA-Graph.
- Grafica.Grafo.Componente: Se encarga de crear una vista del grafo con toda la información referente a su representación gráfica: se definen las formas asociadas a cada objeto del grafo y se ubican dichas formas en un panel que en definitiva será el que visualice el usuario de la aplicación. Para esto también se hace uso de una biblioteca externa denominada JGraph.

A continuación entraremos en los detalles del diseño y la implementación de cada uno de estos subsistemas.

D.3.2.1. Principal

Este conjunto de clases que hemos denominado ‘Principal’ no es un subsistema en sí, sino que simplemente es un paquete que hemos separado para su análisis solo por un tema de claridad. Prácticamente no tiene ninguna lógica y solo contiene las clases para la representación de la ventana principal de la aplicación (que además es el main del programa), la barra de menú que se asoció a la misma, las ventanas internas en las que se visualizan los grafos, y las ventanas para la selección de los archivos GXL que se desean visualizar.

Referente a la selección de archivos (clases *SelectorArchivosAbrir* y *SelectorArchivosGuardar*), se extendió la clase estándar *JFileChooser* para implementar las ventanas de selección utilizando la facilidad de elegir los archivos utilizando el árbol de directorios del sistema (al estilo Explorador de Windows). Además se implementó un flitro de archivos (*FiltroArchivos*) de forma de mostrar solo aquellos con extensión .gxl. Este filtro también se usa en la ventana para guardar los archivos, la cual se implementó de la misma forma que la utilizada para elegir el archivo a abrir.

A continuación presentamos un diagrama de las clases de este paquete:

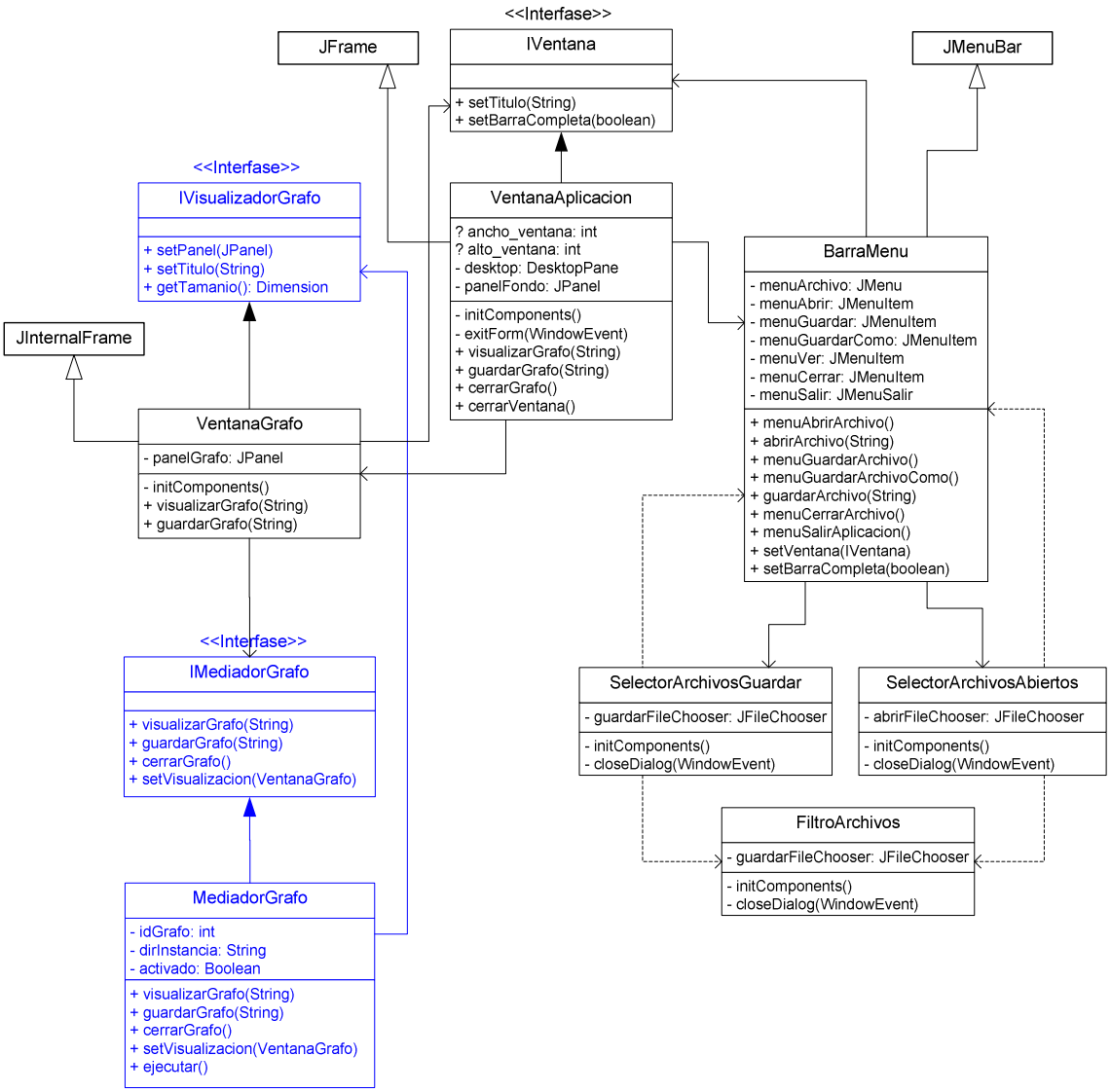


Fig. D.2: Subsistema 'Principal'

Como se ve en la Figura 2, la ventana principal implementa una interfaz (*IVentana*), lo que permite la posibilidad de cambiar la representación gráfica de la ventana sin efectuar demasiados cambios en la aplicación.
También para el caso de la ventana interna para la visualización de los grafos se utiliza una interfaz para el acceso a sus operaciones, lo que genera un grado de independencia importante entre los diferentes subsistemas que definimos. En este caso, el subsistema Grafica.Fabrica, a través de la clase *ManipuladorGrafo,* que, como se verá más adelante, se encarga de obtener un panel cargado con la representación gráfica del grafo y de acceder a la ventana interna para colocar dicho panel. De forma análoga, la comunicación entre *VentanaGrafo* y el subsistema Grafica.Fabrica se realiza a través de la interfaz *IMediadorGrafo*,

la cual brinda las operaciones básicas que podemos realizar a nivel de los archivos de grafos en GXL: abrirlos, guardarlos y cerrarlos.

D.3.2.2. Grafica.Fabrica

La tarea de este subsistema es hacer una mediación entre la información que se obtiene a través de la capa Lógica (y la capa de Acceso de Datos) con todo el subsistema de representación del grafo que se encuentra en la capa de Presentación.
Las operaciones que implementa el subsistema son las básicas sobre los archivos de grafo que mencionamos arriba: abrir, guardar y cerrar. Para abrir un archivo GXL, se obtiene la información de éste, y se pasa la misma a un controlador de grafo en el subsistema Grafica.Grafo.* (específicamente al subsistema Grafo.Grafica.Fabrica); mientras para de guardar la información de un grafo que se está visualizando se sigue el camino inverso: se pide la información al controlador del grafo y se pasa a la capa Lógica.

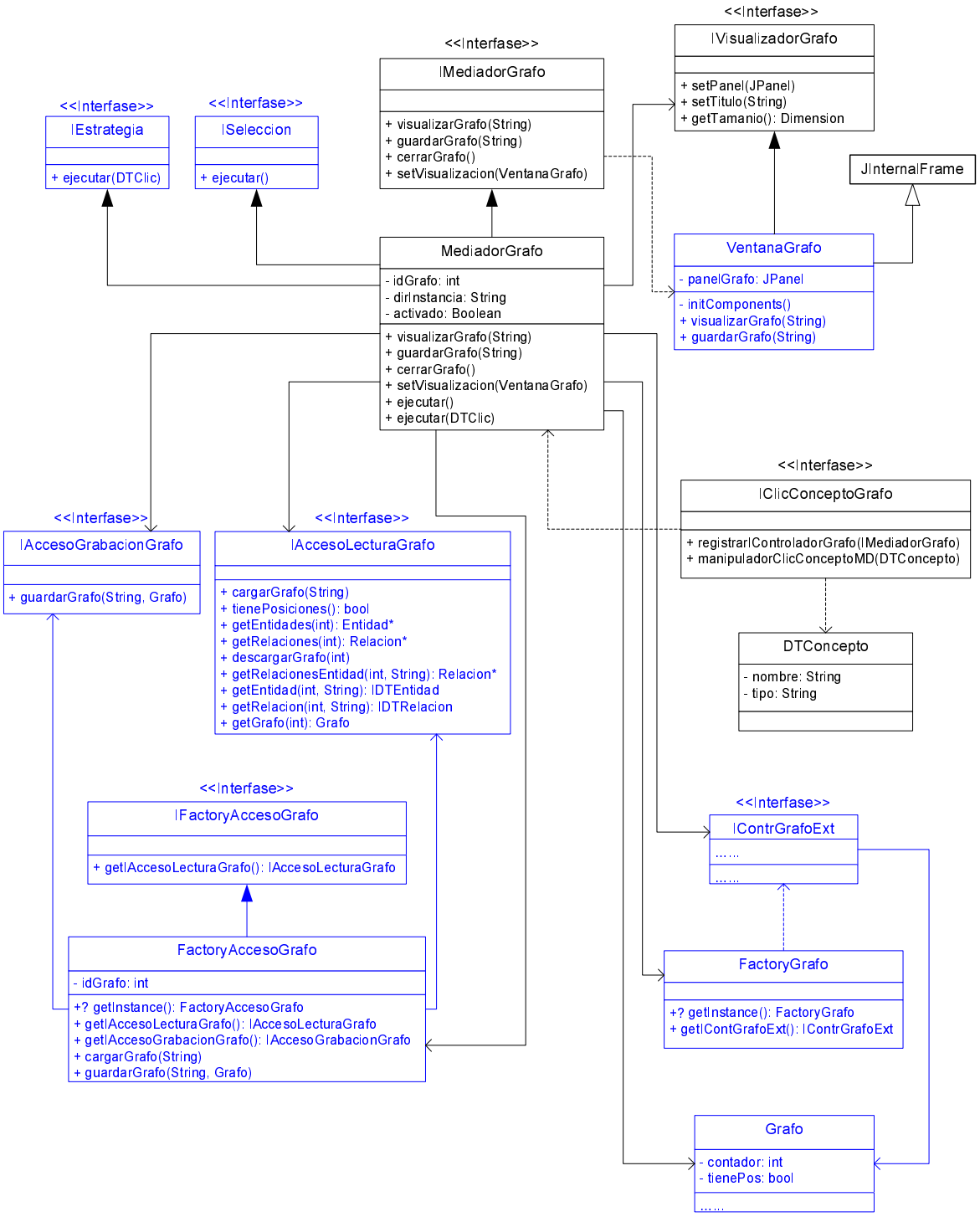


Fig. D.3: Subsistema 'Grafica.Fabrica'

Esta comunicación de datos entre dos subsistemas se realiza a través de Data Types. En este caso el DT utilizado es *Grafo*, el cual permite almacenar la información referente a un grafo, en particular sus nodos y aristas (con los atributos correspondientes a cada uno, modelados en el DT *Atributo*) representados respectivamente por los DT *Entidad* y *Relacion*. Además también es posible almacenar información general del grafo referente a su visualización, como el zoom, el tamaño de la ventana y la utilización de una cuadrícula de fondo.

Al igual que antes toda la comunicación entre los subsistemas se realiza a través de interfaces. Por un lado tenemos la interfaz *IMediadorGrafo* a través de la cual la ventana de visualización del grafo accede a las operaciones de este subsistema. Luego internamente se accede a la información en los archivos de grafos a través de las interfaces *IAccesoLecturaGrafo* e *IAccesoGrabacionGrafo*, las cuales se obtienen de la fábrica de interfaces *FactoryAccesoGrafo*. La existencia de esta fábrica permite controlar que cada una de las interfaces creadas esté asociada a un único archivo GXL, el cual se indica en el constructor de las clases que implementan dichas interfaces; y que solo exista una instancia de cada una de ellas en cada momento, o sea que el sistema solo maneja la visualización de un archivo de grafos en cada instante.
Por otro lado para el acceso al subsistema de representación gráfica se utiliza la interfaz *IContrGrafExt* que brinda las operaciones de acceso para todo el mecanismo de representación del grafo de forma gráfica y también permite la obtención del panel con esta representación grafica que luego será desplegado en la ventana interna del grafo.

También en este subsistema contiene la interfaz de acceso a lo que es el manejo de eventos de usuario sobre la representación del grafo (*IClicConceptoGrafo*), como son los clics sobre sus elementos y el arrastre de los mismos dentro del panel.

D.3.2.3. Grafica.Grafo.Fabrica

Este subsistema es el primero de un conjunto que podría considerarse como un subsistema más general encargado de todo lo necesario para la representación grafica de un grafo, integrado por los subsistemas Grafica.Grafo.* (Grafica.Grafo.Controlador y Grafica.Grafo.Componente además de éste).
La misión de este subsistema es agrupar los diferentes puntos de acceso (o sea, interfaces) para este subsistema general que mencionábamos.

Dado que en sí el subsistema no funciona como tal, ya que no realiza ninguna lógica o actividad, solo podemos agregar una explicación sobre la comunicación de los eventos en la representación gráfica del grafo a lo que serían las clases encargadas de manejarlos.
Estos eventos son detectados por la clase *ManipuladorGrafo* (que en definitiva es el panel en donde está siendo mostrado el grafo) la cual implementa la interfaz *IObservable*. Por otro lado, todas las clases interesadas en ser notificadas de los eventos que se produzcan, deberán implementar la interfaz *IObservador*, la cual contiene todas las operaciones que se notifican desde la representación gráfica; y registrarse también en la interfaz *IObservable* que les interese (en nuestro sistema tenemos solo una, el panel de visualización del grafo como recién mencionamos, pero esto podría extenderse) a través de la operación *agregarObservador()* de la misma. Al momento de en que se produce alguna acción de interés sobre la representación del grafo, se dispara la ejecución de una operación que se encarga de notificar a todos los observadores registrados sobre la ocurrencia del evento.
Con este mecanismo de manejo de los eventos en el grafo, es posible una extensión sencilla de los eventos que se pueden manejar (solo habría que incluirlos en una clase que implemente *IObservable*) o de las acciones que se toman en caso de un evento (solo habría que incluirlos en una clase que implemente *IObservador*)

Habiendo descripto el funcionamiento interno de este subsistema, sólo resta mostrar el diagrama de clases asociado al mismo:

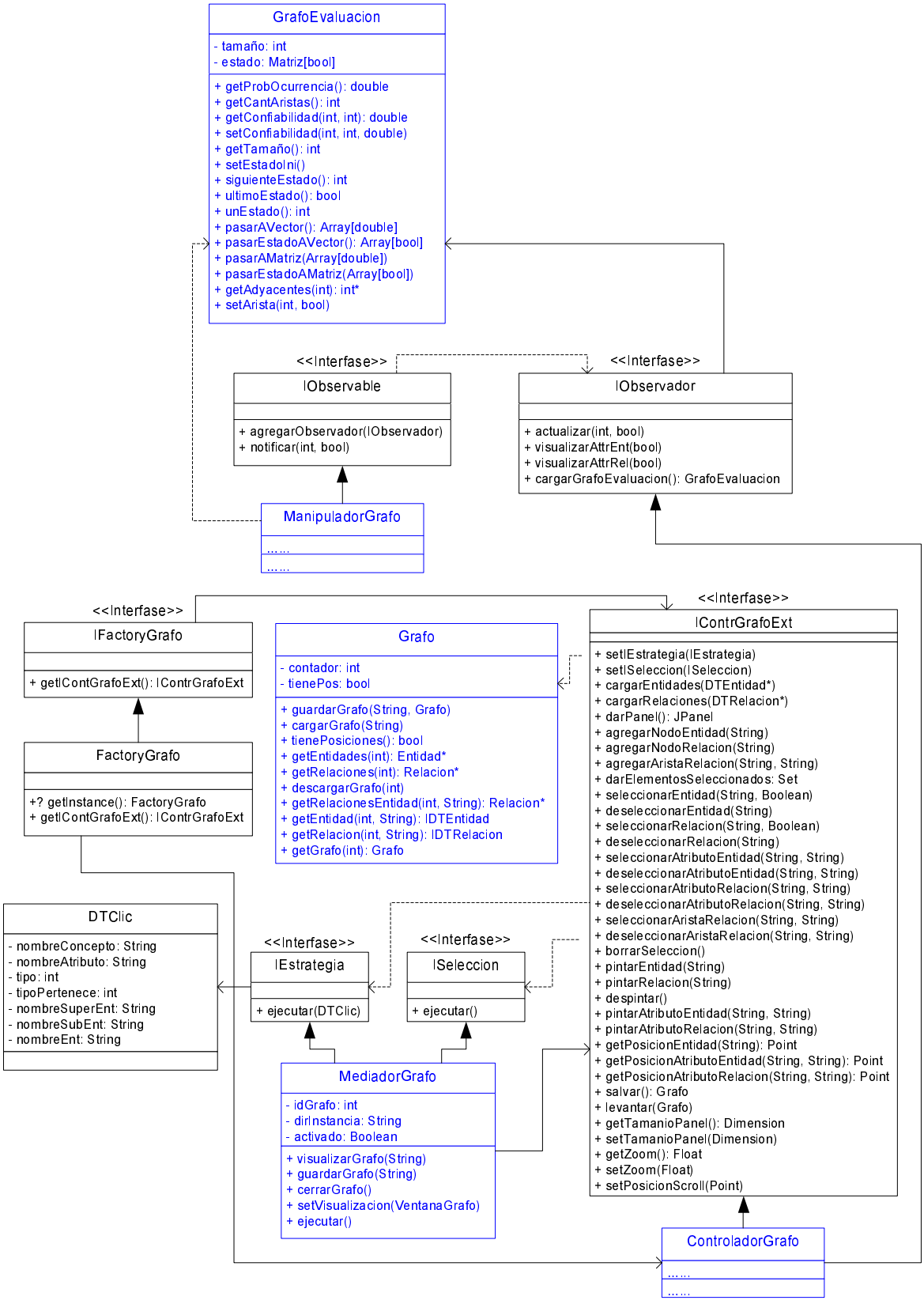


Fig. D.4: Subsistema 'Grafica.Grafo.Fabrica'

D.3.2.4. Grafica.Grafo.Controlador

Entrando definitivamente en lo que es la representación en sí del grafo, llegamos a este subsistema encargado de crear una representación en memoria de la estructura del grafo con la información referente a la estructura del mismo (nodos, aristas, atributos de éstos y conexiones entre ellos), al posicionamiento de los diferentes elementos, y el estado de los mismos referente a su selección (seleccionados o no).
La información referente a los elementos que componen el grafo y las conexiones entre ellos se almacena en diferentes conjuntos (implementados como hashs) específicos para cada una de las funcionalidades en la clase *ControladorGrafo* que implementa la interfaz *IContrGrafExt*. Esta última, como habíamos mencionado antes, provee el punto de acceso a las operaciones básicas del grafo a los demás subsistemas. Como decíamos, esta información básica del grafo no se almacena en una estructura de grafo sino que se almacena en colecciones particulares de la clase que podrían cambiarse en otra implementación de la interfaz.
Para las demás operaciones (de posicionamiento y de selección) el controlador del grafo solamente presenta su acceso, siendo implementadas en lo que es el esquema del grafo (ver descripción a continuación de la Figura 5).

A continuación presentamos el diagrama de clases de lo que sería esta primera subcapa dentro de este subsistema y luego describiremos la funcionalidad de la subcapa encargada del manejo del esquema del grafo:

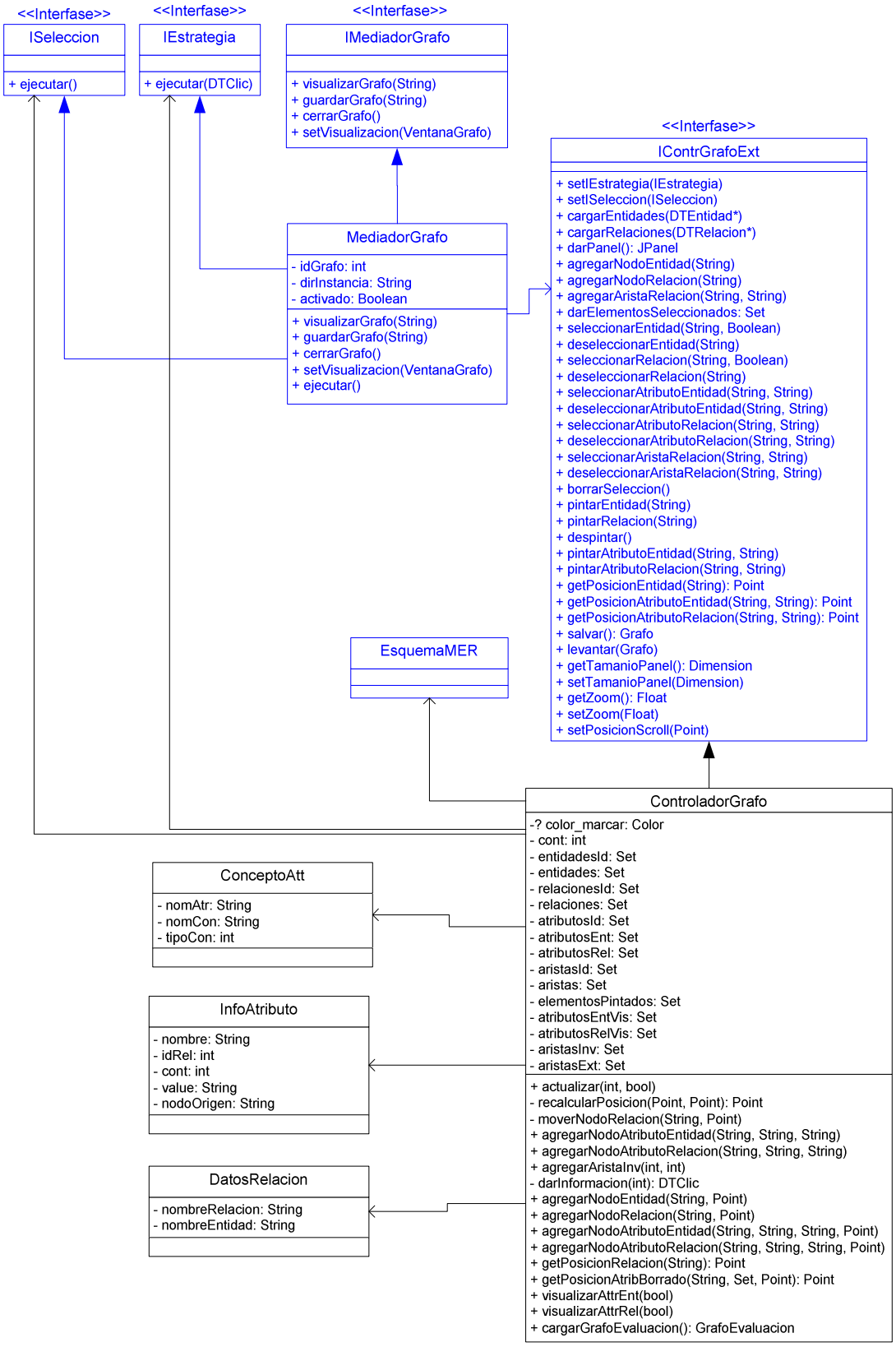


Fig. D.5: Subsistema 'Grafica.Grafo.Controlador'

Internamente además de cargar todas estas colecciones de información, el controlador del grafo se encarga de crear lo que llamamos el esquema del grafo: una representación del mismo a través de clases que extienden a clases de la biblioteca SDA-Graph y que en este

caso sí siguen una estructura de grafo. En definitiva el esquema maneja la información sobre el posicionamiento de los elementos en la pantalla y la selección de los mismos.
Cabe aclarar que a pesar de que aquí se mantenga una estructura con la información del grafo e incluso sus posiciones, esta estructura no es la que se utiliza para la representación gráfica del mismo sino que es una representación lógica del grafo, aunque sí está estrechamente relacionada con la estructura que se utiliza para la representación gráfica del mismo que se encuentra en el subsistema Grafica.Grafo.Componente (clase *ManipuladorGrafo*).

El punto de acceso al esquema del grafo es la clase *EsquemaMER*, la cual contiene lo que sería la representación del grafo particular en un *EsquemaGrafo* (la cual extiende a la clase *MultiGraph* de la biblioteca SDA-Graph) en el que están definidos los nodos y aristas del grafo a través de las clases *EsquemaNodo* y *EsquemaArista* respectivamente.

En este punto cabe mencionar los objetos que manejaremos como elementos del grafo y la nomenclatura que utilizaremos para referirnos a ellos.
Conceptualmente los grafos con los que trabajaremos son grafos no dirigidos de la forma G=(V,E) en donde V es el conjunto de nodos y E es el conjunto de aristas de la forma $k(e_1, e_2)$ en donde e_1 y e_2 pertenecen a E y son diferentes entre sí.
Teniendo en cuenta que es necesario representar atributos asociados a las aristas y que tanto la representación de atributos como la asociación de los mismos a las aristas no están contemplados como construcciones básicas de las bibliotecas que utilizamos, definimos el siguiente sistema para representar los grafos con los que trabajamos:

- Tanto los nodos como las aristas y los atributos del mismo estarán representados por subclases de *EsquemaNodo*: *ClaseNodo*, *RelacionNodo*, *AtributoNodo* respectivamente.
- Los conectores que unen tanto los *ClaseNodo* con los *RelacionNodo* (que representarían las aristas del grafo) como éstos con los *AtributoNodo* (que cumplen la función de unir los atributos con el nodo o arista que corresponda), se representan respectivamente con las clases *RelacionArista* y *AtributoArista*, ambas subclases de *EsquemaArista*.

Otro punto a mencionar es el referente al algoritmo de ordenación implementado para la distribución de los diferentes objetos de forma relativamente armónica en la pantalla.
El mismo se basa en el posicionamiento de los nodos en la pantalla en niveles, construyendo un árbol de cubrimiento para el grafo usando un algoritmo BFS.
Para cada nodo se buscan todos sus adyacentes y se colocan en un nivel inferior, repitiéndose esta distribución para cada uno de ellos en un orden determinado. Finalmente se agregan las aristas que quedaron excluidas al momento de la construcción del árbol.

Para culminar con esta subsección presentamos el diagrama de clases de la parte referente al esquema del grafo:

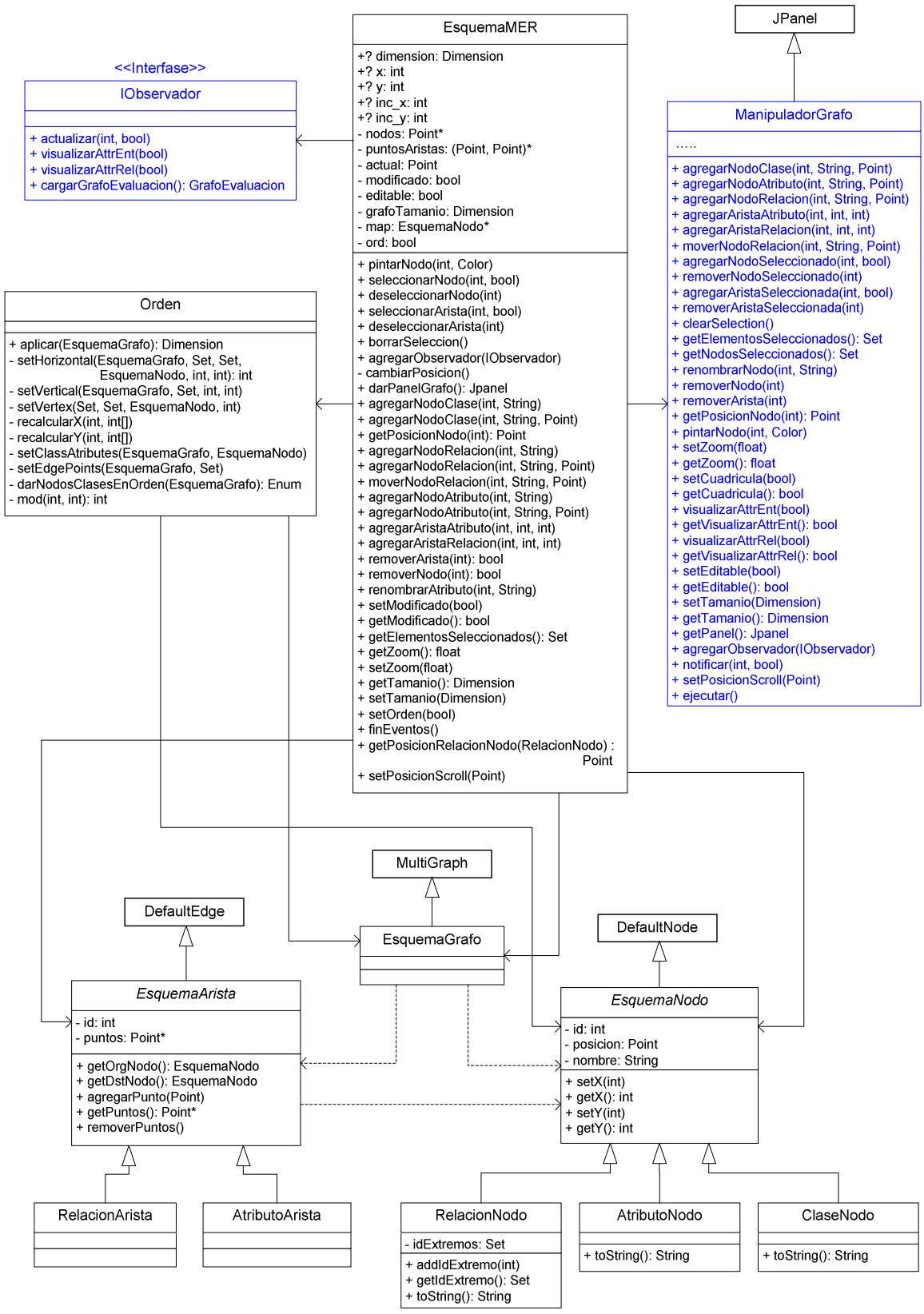


Fig. D.6: Subsitema 'Grafica.Grafo.Controlador' – Esquema del grafo

D.3.2.5. Grafica.Grafo.Componente

El último subsistema (como veremos adelante la denominación subsistema en este caso es bastante artificial y se ha usado simplemente para mantener la nomenclatura que venimos

manejando; sería más correcto hablar de paquete) que hemos dejado para desarrollar de la capa de Presentación, consiste justamente del que se encarga de la representación del grafo en la pantalla. Para eso nos basamos en otra biblioteca externa que precisamente realiza esta función y se denomina JGraph. Una de sus principales ventajas es que trabaja íntimamente relacionada con la biblioteca SDA-Graph (de hecho son productos de la misma organización) que hemos usado como base para la representación de la estructura del grafo en memoria.
La otra gran ventaja de esta biblioteca es la facilidad de extensión de las clases que brinda para la adaptación de su funcionalidad al caso de aplicación particular. A partir de las características que comentamos en la construcción de nuestro esquema de grafo, podemos diseñar una representación gráfica del mismo con las siguientes características:

- Tres formas de entidades:
  - Cuadrados para representar a los nodos del grafo.
  - Círculos para representar los atributos tanto de aristas como de nodos del grafo.
  - Puntos para representar las aristas.
- Una forma para las relaciones (la clásica, como una línea recta que une los elementos a relacionar) para representar los dos tipos de aristas que se pueden presentar:
  - Entre la entidad que representa una arista y la entidad que representa uno de los nodos extremo de la misma.
  - Entre las entidades que representan aristas o nodos y las entidades que representan sus correspondientes atributos.

La clase principal de este subsistema es *ManipuladorGrafo* la cual, además de manejar todo lo necesario para la representación gráfica del grafo, es un panel (extiende la clase *JPanel*) en cual se dibujará el grafo y luego será pasado hacia la ventana interna de visualización ubicada dentro de la ventana principal de la aplicación. Aprovechando esta posibilidad, en la misma clase se incluyó una barra de botones con las operaciones básicas sobre el grafo que se dibuja (zoom, visualización de atributos de nodos y aristas, visualización de una cuadrícula de fondo) y en la que aprovechamos para incluir un botón para hacer el llamado a la aplicación de evaluación de grafos adaptada para tomar como entrada el grafo con que se está trabajando.
A pesar de tratarse del punto de acceso al subsistema (en nuestra implementación particular desde el subsistema Grafica.Grafo.Controlador), la clase *ManipuladorGrafo* no presenta ninguna interfaz para el encapsulamiento de su implementación. Esto se debe a que lógicamente este paquete no es un subsistema, sino que forma parte de lo que es precisamente el subsistema Grafica.Grafo.Controlador. Conceptualmente, el manejo de la representación del grafo debe mantener una consistencia entre como se manejan las clases en memoria y como se muestran en la ventana, de forma que se tomó la decisión de contar con una sola interfaz de acceso (que en nuestro caso sería *IContrGrafExt*) y luego mantener la consistencia haciendo que cada operación de la interfaz (y por ende de *ControladorGrafo*) esté reflejada en la clase encargada de manejar la gráfica del grafo, o sea, en *ManipuladorGrafo*.
Luego, mediante el sistema de clases observadoras y clases observadas, que describimos oportunamente en la subsección referente al subsistema Grafica.Grafo.Fabrica, las acciones que efectúe el usuario sobre el grafo se propagarán hacia la representación de su estructura en memoria, manteniéndose la consistencia ahora en el otro sentido.

La relación de la clase *ManipuladorGrafo* con otras clases externas a este paquete, queda representada en el siguiente diagrama de clases resumido:

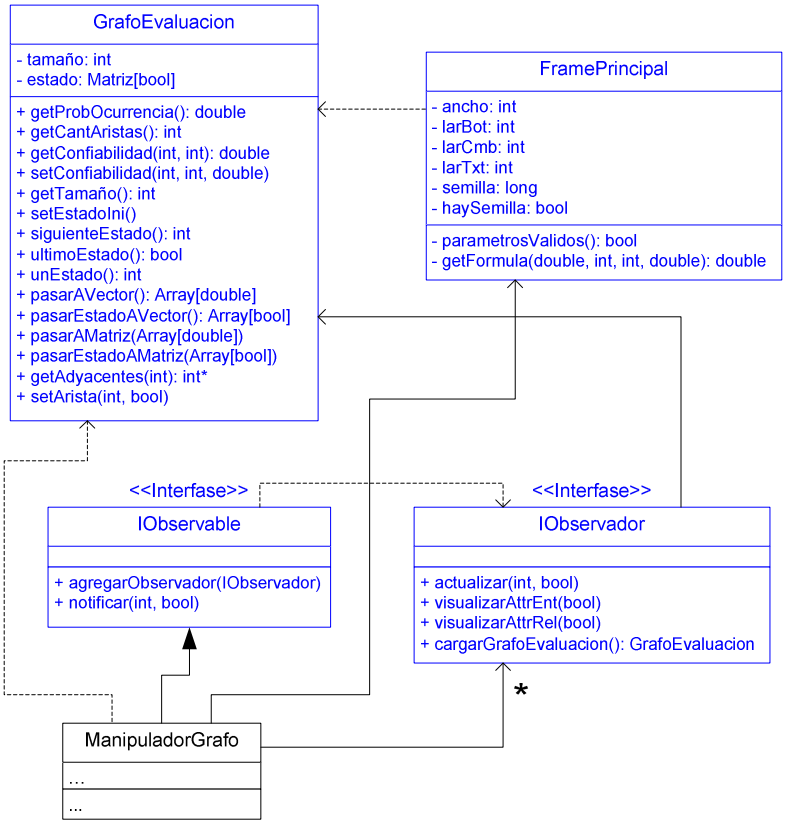


Fig. D.7: Interacción de *ManipuladorGrafo* con otros paquetes

Además de lo que es *ManipuladorGrafo*, el subsistema cuenta con una serie de clases auxiliares a lo que es la función de éste.
La clase *MyJGraph* concentra toda la información del grafo: al momento de agregar un nodo o arista al grafo, el mismo se agrega en la instancia de *MyJGraph* que tiene asociada el manipulador (o sea, se agrega al *JGraph* que extendemos). La única extensión de la lógica del objeto estándar *JGraph* que hacemos es la referente a las formas que asociaremos a cada uno de estos elementos que se agregan al grafo, para lo cual sobrescribimos las operaciones 'createVertexView' y 'createEdgeView' de la clase *JGraph*, de forma de devolver instancias de las clases de vistas (todas ellas extensiones de *VertexView* y *EdgeView*) que hemos creado. Para cada una de estas clases de vistas "propias", la extensión con respecto a la funcionalidad estándar que hemos hecho corresponde con los renderers que tienen asociado cada una. Dichos renderers en definitiva son los que contienen la forma de dibujar el elemento en el panel de visualización del grafo.

En resumen, todas estas clases auxiliares en el subsistema tienen como función extender las clases que brinda JGraph para adaptarlas a nuestra realidad específica, lo que queda claro en el siguiente diagrama de clases completo del paquete:

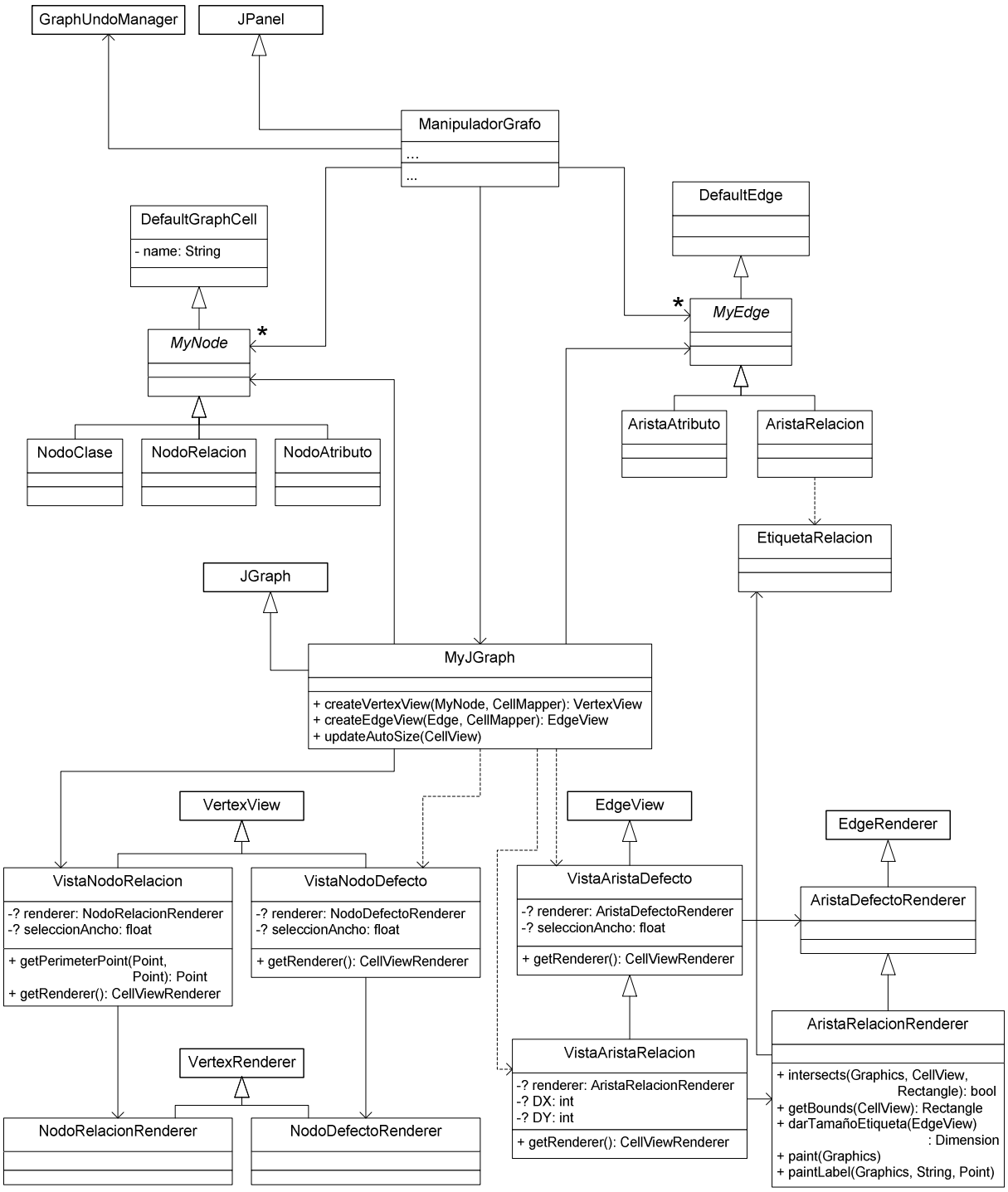


Fig. D.8: Subsistema 'Grafica.Grafo.Componente'

Dejamos para el final la presentación de lo que sería el diagrama de clases de *ManipuladorGrafo*, que por razones de espacio no incluimos en los diagramas precedentes:

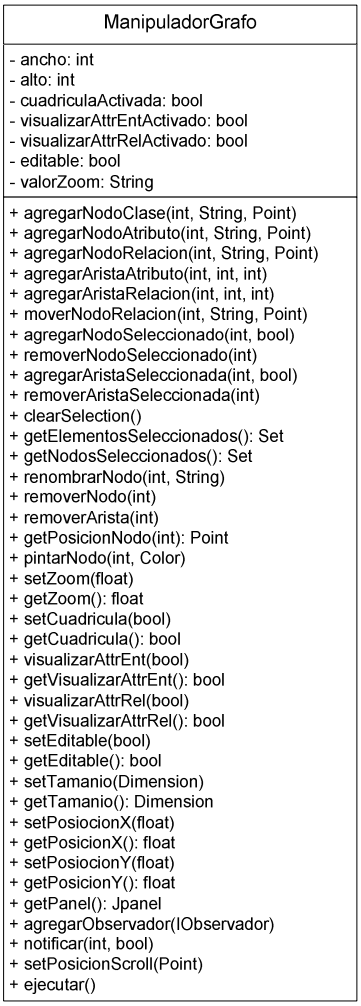


Fig. D.9: Diagrama de clases de *ManipuladorGrafo*

D.3.3. Capa Lógica

Como se mencionó al comienzo de la sección de diseño, la capa Lógica solamente contiene la lógica para la ejecución de los algoritmos de evaluación sobre el grafo que se está visualizando y una fábrica para el acceso a las interfaces de la capa de Acceso a Datos.

Dado que el diseño de la herramienta de Evaluación se describe en detalle en el documento correspondiente (**Apéndice E**: Evaluador), solo nos queda mencionar algunas particularidades de lo que es la adaptación de dicha herramienta para tomar como entrada el grafo que se esté visualizando.
Para empezar, la ejecución del sistema de evaluación se dispara desde un botón ubicado en la barra de botones que se añadió al panel de visualización del grafo (clase *ManipuladorGrafo*) y el evento que maneja la acción de presionar este botón también se encuentra allí. Dada la situación de que el usuario presione el mencionado botón, el sistema debe almacenar la información del grafo en el formato esperado por el Evaluador, acción que se realiza guardando la información del grafo en un data type *GrafoEvaluacion*.
Una vez creado este data type es pasado a la ventana de parámetros del Evaluador, la cual es una versión reducida de la ventana original en la que solo es necesario ingresar el tiempo de vida de los paquetes en la red (D) y la cantidad de pedidos por nodo (C). Al igual que en la herramienta de Evaluación, se podrán elegir tanto los algoritmos que se desean ejecutar como

los resultados que se quieren visualizar; pero no está disponible para elegir el lenguaje para los algoritmos, ejecutándose las versiones de los mismos implementadas en Java.
El resto de los parámetros necesarios para la evaluación (cantidad de nodos N, y cantidad de conexiones promedio por nodo CCP) serán obtenidos de la topología del grafo que se pretende evaluar. Para éste último parámetro que mencionamos surge una restricción importante para lo que es el grafo: todas sus aristas deben tener definido un atributo de nombre “Confiabilidad” que indique precisamente la probabilidad de que dicha arista esté activa en un cierto momento.

Para lo que es el resto de la capa Lógica, la fábrica *FactoryAccesoGrafo* se encarga de obtener y retornar las interfaces para la lectura y grabación (*IAccesoLecturaGrafo* e *IAccesoGrabacionGrafo* respectivamente) de la información de un grafo contenida en un archivo GXL, que se implementan en la capa de Acceso a Datos, la cual se describe a continuación.

D.3.4. Capa de Acceso a Datos

Esta capa es la encargada de la interacción con los archivos GXL que servirán de fuente de datos para los grafos representables por la aplicación.

Como se acaba de mencionar arriba, el acceso a esta capa se da a través de dos interfaces: *IAccesoLecturaGrafo* para las operaciones de lectura de un archivo GXL e *IAccesoGrabacionGrafo* para las correspondientes a la escritura de dichos archivos. Ambas son implementadas por una misma clase (*AccesoGrafo*) que funciona como controlador del acceso a los archivos, teniendo además la potencialidad de manejar más de un grafo al mismo tiempo a través de la asociación de un identificador único a cada uno de ellos para su almacenamiento en una colección propia. Esta funcionalidad no es explotada en nuestro sistema pero se mantuvo para una eventual extensión del mismo en este sentido.
Lo que es el acceso en sí a los archivos GXL se realiza, a través de la mencionada clase *AccesoGrafo*, en la clase *ModeloGrafo*. Ésta hace uso de la biblioteca Gxl [HSEW 02] para el manejo de archivos homónimos, para el acceso a la información en el archivo y su posterior almacenamiento en colecciones de nodos y aristas propias de la clase.
El almacenamiento y posterior comunicación de esta información al resto del sistema, se efectúa a través de un conjunto de data types colocados en el paquete AccesoDatos.DT, que también sirven de nexo entre la capa que describimos y la Lógica.

Además de lo que es la información del grafo en sí mismo, se maneja también información general referente el grafo y al estado de su visualización, como es:

- El valor del zoom con que se está visualizando o se desea visualizar el grafo.
- Si la cuadricula de fondo está activa o no.
- El tamaño del panel en el que se está visualizando o se desea visualizar el grafo.
- Las posiciones de cada uno de los nodos y aristas.

Para el almacenamiento de las tres primeras propiedades, se definieron atributos globales del grafo en el archivo GXL, mientras que para la última se definieron dos atributos específicos de cada nodo o arista con los valores de su coordenada en el panel de visualización. La utilización de estos atributos específicos con fines propios de la aplicación, genera otra restricción en el grafo ya que el mismo no podrá tener un atributo con el mismo nombre que hemos tomado para guardar estos datos. Asumimos que esta restricción no es muy grave, ya que los nombres elegidos (strings de la forma “@NOMBRE_ATRIBUTO@”) parecen poco factibles de ser encontrados en los grafos que se estima van a ser normalmente utilizados.

El diagrama de clases para la misma queda como se muestra a continuación:

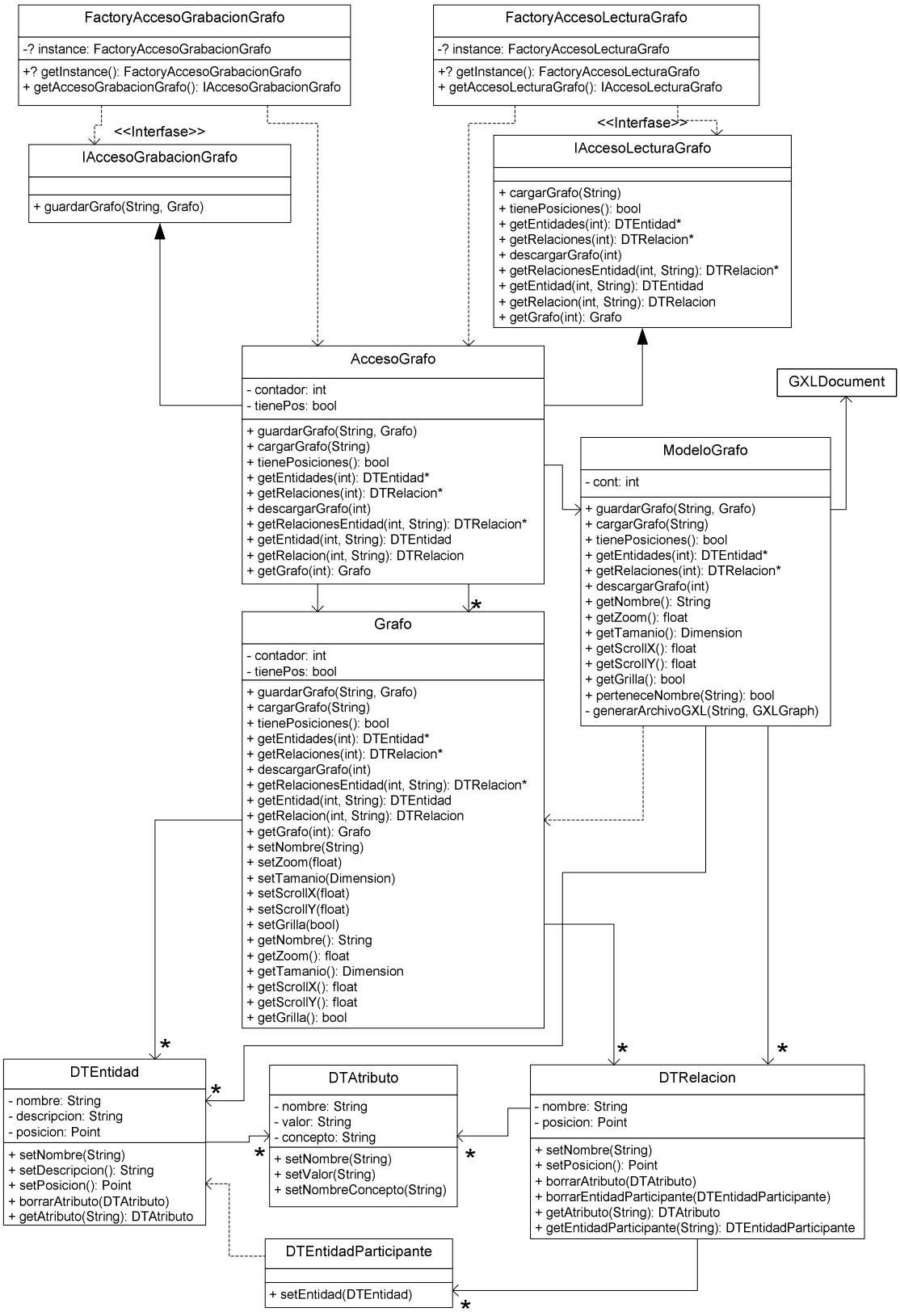


Fig. D.10: Capa de Acceso a Datos

D.4. Descripción de la arquitectura

En la presente sección nos proponemos hacer una breve descripción de la arquitectura del sistema desde la perspectiva de un análisis de los casos de usos más importantes que se puedan detectar sobre el mismo.

Para eso comenzaremos por definir los casos de usos que a nuestro interés resultan más importantes, de forma de no extendernos demasiado con el análisis. Los dos casos de uso que se considerarán, constan de la apertura y visualización de un grafo y del almacenamiento de la información de un grafo que se está visualizando.
A partir de estos casos de uso describiremos su trazabilidad con el modelo de diseño de la aplicación, y a partir de la trazabilidad de los subsistemas encontrados en este último llegaremos al modelo de implementación.

D.4.1. Diagrama de casos de uso relevantes a la arquitectura

Desde una perspectiva de los casos de uso de la aplicación, se pueden diferenciar dos casos básicos: la visualización de un grafo desde un archivo GXL y la grabación de los cambios efectuados sobre el mismo. Un tercer caso de uso de interés podría ser la evaluación de un grafo que se esté visualizando, pero dado que este caso impacta principalmente sobre lo que sería la arquitectura de la aplicación de evaluación de grafos, el mismo será omitido para este análisis simplificado de la arquitectura del visualizador.

A continuación presentamos el diagrama de casos de uso para estos dos casos que desarrollaremos en esta sección.

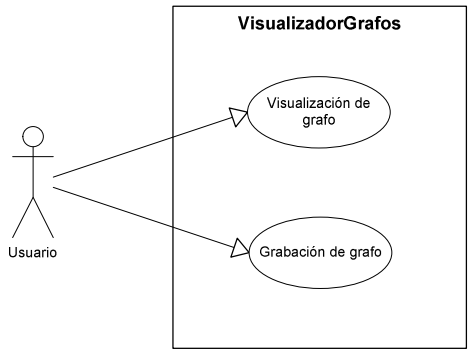


Fig. D.11: Diagrama de casos de uso

D.4.2. Trazabilidad del Modelo de Casos de Uso al Modelo de Diseño

A partir de lo que son las acciones realizadas por el usuario sobre el sistema, que se presentan en el modelo de casos de uso resumido que presentamos arriba, el siguiente paso es detectar la trazabilidad de las mismas sobre el modelo de diseño que presentamos en la sección 'Consideraciones generales de diseño'.

Para los casos de uso que trataremos, la misma sería como sigue (dado que la trazabilidad es la misma para los dos casos de uso que planteamos, nos hemos tomado la libertad de representarlos en el mismo diagrama):

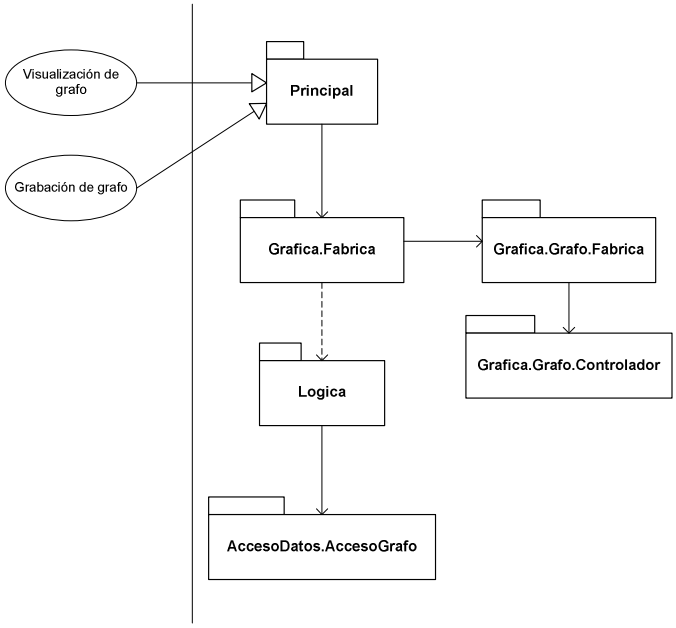


Fig. D.12: Trazabilidad del modelo de casos de uso al modelo de diseño

La descripción de lo que serían los subsistemas fue realizada a lo largo de las secciones previas del presente documento, por lo que no se repetirá aquí.

D.4.3. Trazabilidad del Modelo de Diseño de Uso al Modelo de Implementación

Dado que para el análisis de los subsistemas los mismos fueron mapeados uno a uno con los diferentes paquetes que se implementan, la trazabilidad del modelo de diseño al modelo de implementación es automática.
Tampoco hay comentarios para hacer referentes a los diferentes paquetes del sistema, ya que los mismos fueron intercalados en los comentarios que se realizaron respecto a los subsistemas en las secciones anteriores del documento.

D.4.4. Diagramas de secuencia

A partir de lo que son los subsistemas (o paquetes) involucrados en la implementación de los casos de uso presentados, podemos refinar más nuestro análisis de los mismos mostrado un diseño concreto para las operaciones que implementan dichos casos.
Dos opciones para este análisis serían la realización de Diagramas de Colaboración que muestren la interacción entre las clases del sistema para llevar a cabo las operaciones; y la

realización de Diagramas de Secuencia que permitan ver precisamente la secuencia de llamados que se realizan en el sistema para la implementación de las operaciones. Para los casos particulares que estamos desarrollando, nos pareció una mejor opción, tanto por claridad como por facilidad de realización, la descripción mediante Diagramas de Secuencia de las operaciones 'menuAbrirArchivo' y 'menuGuardarArchivo'. Los mismos se presentan a continuación:

D.4.4.1. menuAbrirArchivo

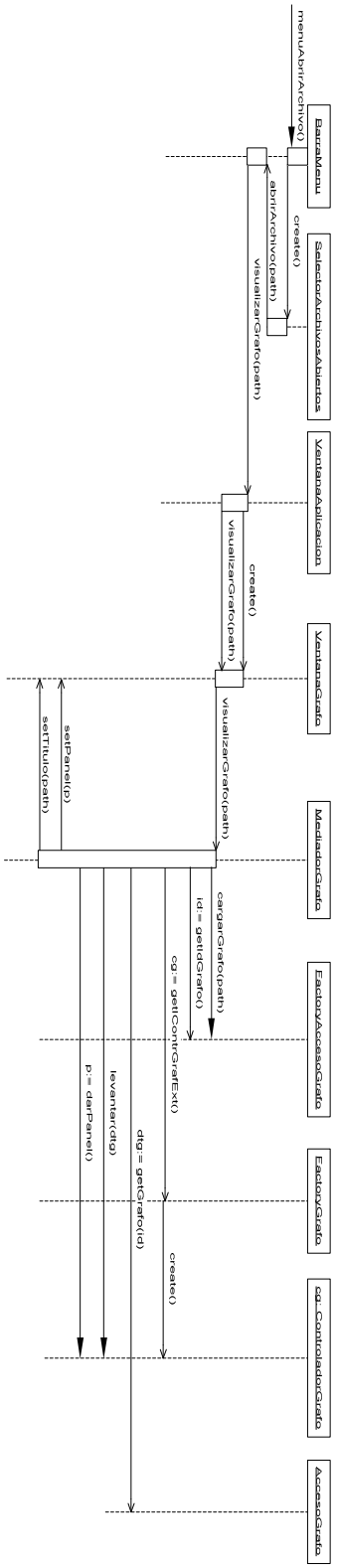


Fig. D.13a: Diagrama de secuencia 'menuAbrirArchivo'

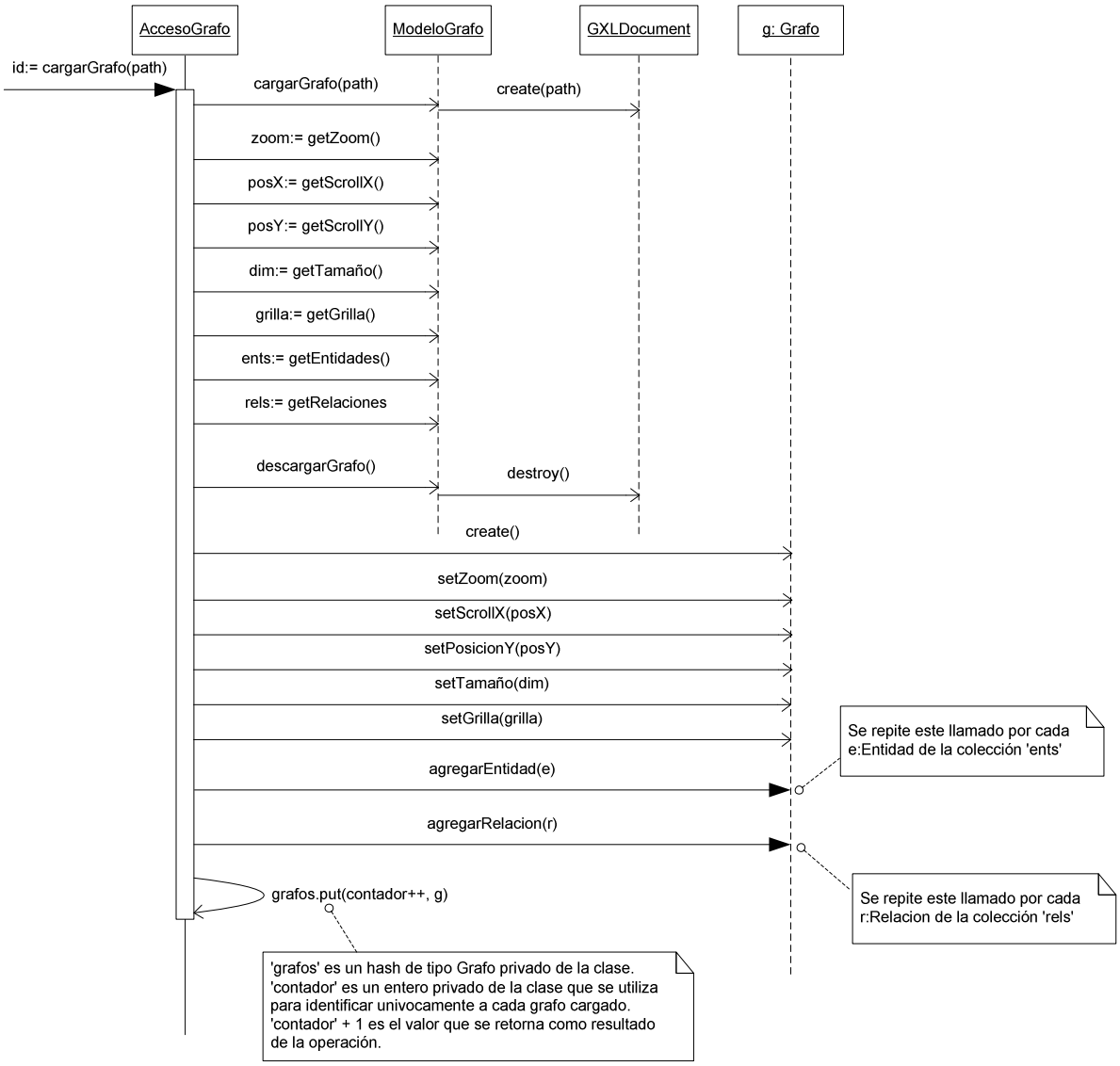


Fig. D.14b: Diagrama de secuencia 'menuAbrirArchivo'

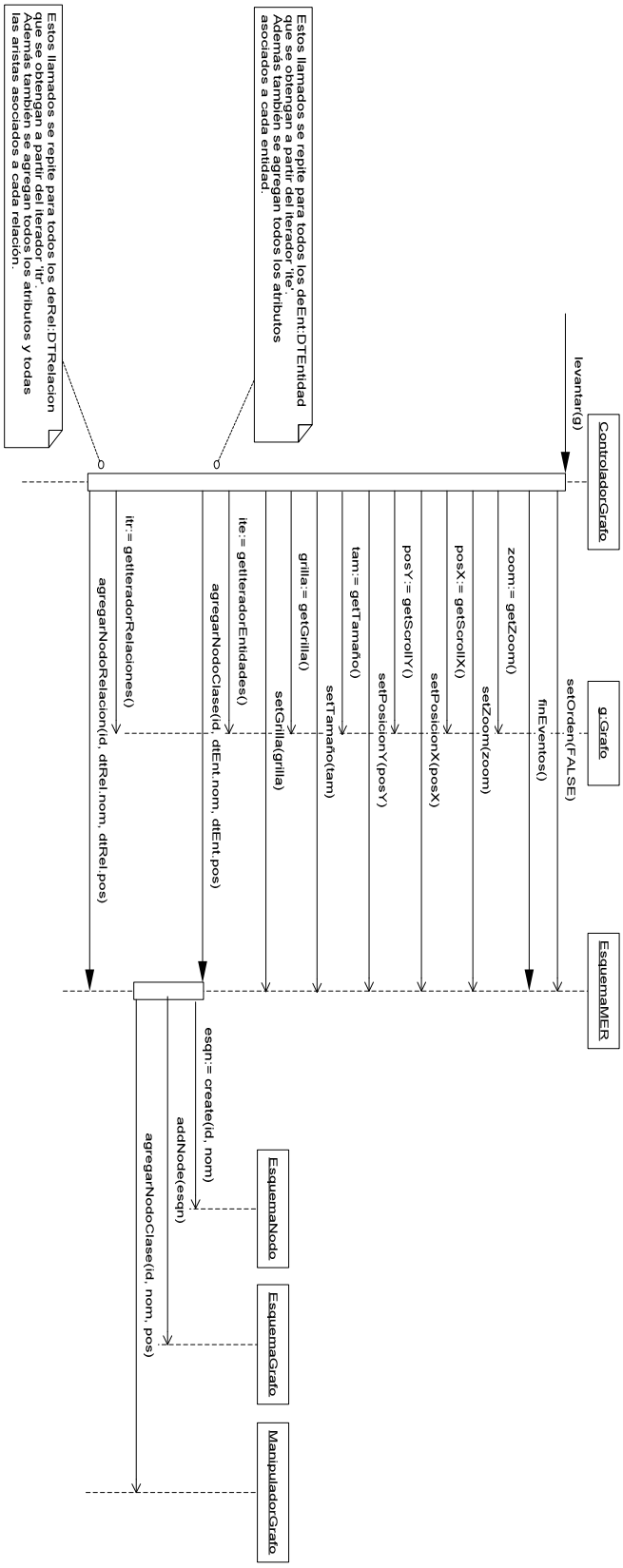


Fig. D.15: Diagrama de secuencia 'menuAbrirArchivo'

D.4.4.2. menuGuardarArchivo

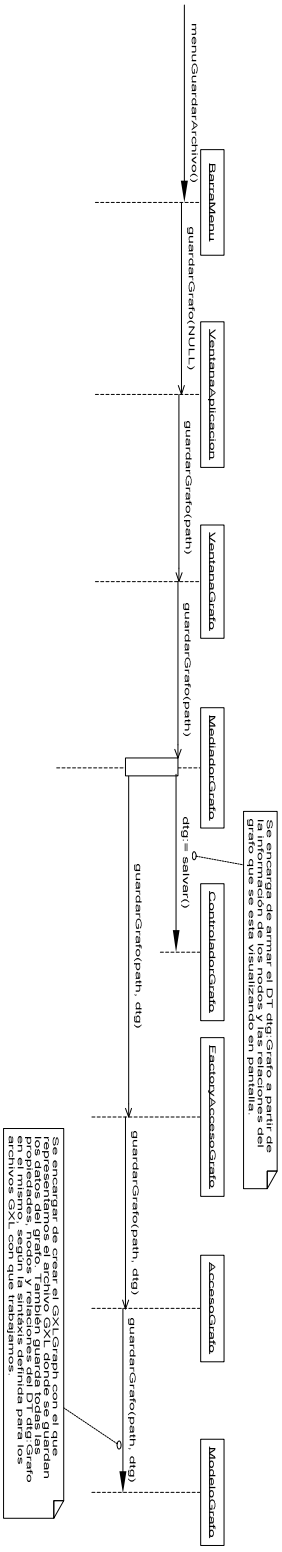


Fig. D.16: Diagrama de secuencia 'menuGuardarArchivo'

D.5. Prueba de validación

Para la herramienta de visualización de grafos solo se efectuaron pruebas unitarias como etapa de testeo de la aplicación una vez estabilizada su implementación. Las mismas permitieron la detección de un número importante de bugs de la aplicación, muchos de los cuales fueron solucionados en la versión que se entrega. Por una lista de bugs que no fueron solucionados, referirse a la sección D.6.2 del presente documento.

Debido a la complejidad de la aplicación y la consiguiente complejidad para el planteo y realización de una prueba de validación formal de la misma, ésta fue dejada fuera del presente proyecto quedando si como un trabajo de extensión futuro para el mismo.

D.6. Trabajo futuro

D.6.1. Extensiones

La herramienta de visualización de grafos representados en GXL presenta algunos puntos de extensión que pueden resultar de interés para la realización de una nueva versión de la misma:

- Incluir la posibilidad de manejar otros tipos de archivos además de GXL:
  Habría que crear un módulo encargado de leer archivos del nuevo formato y almacenar su información en un data type *Grafo*, además de hacer los arreglos necesarios para dar la posibilidad al usuario de seleccionar archivos con este nuevo formato.

- Extender la herramienta para poder crear y modificar grafos:
  Se podrían crear opciones para agregar componentes al grafo a través de la interfaz gráfica y luego almacenar las mismas en un archivo determinado.

- Permitir la visualización de más de un grafo al mismo tiempo:
  Para esta extensión habría que sacar la calidad de 'Singleton' a algunas clases involucradas al manejo de la representación en memoria y gráfica de los grafos, además de permitir múltiples instancias de ventanas internas para la visualización de los mismos, y de implementar algún mecanismo para la identificación de los mismos a nivel de su representación interna.

- Ampliar los tipos de grafos válidos para la aplicación:
  Dado que para el desarrollo del proyecto fue suficiente una herramienta para la visualización de grafos simples, en pos de extender la misma más allá de los límites impuestos para este trabajo, se podría considerar la posibilidad de incluir grafos dirigidos, multigrafos, atributos de tipos compuestos, etc.

- Mejoras en el algoritmo de ordenación de los objetos en la ventana:
  Se podrían buscar otros algoritmos para la representación de los elementos gráficos en la ventana, principalmente los atributos de nodos y aristas.

D.6.2. Bugs

Hasta el momento durante las pruebas realizadas con la aplicación se han detectado algunos errores o bugs que debido a los tiempos de desarrollo planteados para el proyecto quedarán sin solucionar en esta primera versión de la aplicación:

- Scroll horizontal en la ventana interna de visualización de grafos:
  Estando visualizando un grafo, si el usuario selecciona un nodo y lo mueve fuera del área de visualización de la ventana, el mismo queda inaccesible para ser movido nuevamente al área de visualización, a menos que se disminuya el zoom. Aparentemente este es un problema en la actualización del scroll horizontal de la ventana interna, pero el mismo no pudo ser solucionado.

- Los textos de los nodos y atributos están limitados a no contener tildes, letras ñ, ni caracteres especiales.

- Los bloques en que se visualizan los atributos son de tamaño constante, por lo que en algunos casos en que el nombre de alguno sea demasiado extenso, el mismo puede quedar cortado en la visualización.

- Al estar visualizando un grafo y seleccionar la opción de menú 'Archivo→ Guardar como' el grafo se guarda correctamente con el nuevo nombre indicado por el usuario, pero en la ventana de visualización se mantiene abierto el grafo original, en lugar de cambiarse al nuevo grafo guardado.

D.7. Referencias

[JGp 00]
JGraph Ltda.
http://www.jgraph.com/
http://sourceforge.net/projects/jgraph/

[HSEW 02]
Graph Exchange Lenguaje.
Ric Holt, Andy Schürr, Susan Elliott Sim y Andreas Winter.
http://www.gupro.de/GXL, Julio de 2002.

[PIS 02]
Proyecto de Ingeniería de Software: Herramienta 'Sim Engine'
Varios autores. Tutor: Regina Motz.
Informe interno, Facultad de Ingeniería, Universidad de la República, Uruguay, 2002.

[PIS 03]
Proyecto de Ingeniería de Software: Sistema editor de metadatas 'EDM5'
Varios autores. Tutor: Fernando Carpani
Informe interno, Facultad de Ingeniería, Universidad de la República, Uruguay, 2003.

[SDA 00]
http://www.dc.teknowledge.com/external/sda-graph/

:: Apéndice E

:: Evaluador: CDA Evaluator

E.1. Introducción

En los apéndices C y D presentamos respectivamente dos aplicaciones (Visualizador de Grafos y Traductor HEIDI-GXL) de alguna forma complementarias o auxiliares a lo que es el objetivo medular del Proyecto, sirviendo las mismas como forma de interfaz para el ingreso de instancias (problemas específicos) a lo que sería nuestro motor de cálculo, pensado para trabajar sobre un clase de problemas definida por nuestro modelo de estudio.

En este apéndice comenzaremos a describir precisamente las dos partes de este motor de cálculo sobre nuestro modelo, dividiendo el análisis básicamente en 3 partes: la presentación formal del los métodos de evaluación, el desarrollo del diseño de la aplicación, y el análisis de los resultados.

Para empezar incluiremos una sección en donde presentamos el modelo con que trabajaremos y la batería de métodos que culminaremos implementado.

Seguidamente desarrollaremos lo que es la aplicación de evaluación de grafos.
Concretamente, el evaluador es una aplicación que, tomando como entrada un grafo completo dado por la cantidad de nodos *N* más una serie de parámetros globales al mismo, a saber, la cota para el diámetro de los caminos válidos (D) y la cantidad promedio de pedidos por nodos (CCP); realiza la evaluación de tres propiedades del mismo: la confiabilidad diámetro acotada, el porcentaje de nodos comunicados por caminos de largo menor que D, y la carga de paquetes circulando en la red.

Para enriquecer lo que será el análisis de resultados, el cálculo de los parámetros de retorno se realizará mediante 3 algoritmos diferentes: generando todos los estados del grafo de entrada (variante que llamaremos Generación Completa de Estados o simplemente GCE), utilizando el método de Monte Carlo estándar (o Monte Carlo Crudo, abreviado por MCC), y finalmente aplicando un método de reducción de varianza sobre la base del método de Monte Carlo estándar denominado Monte Carlo Antitético generalizado (o MCA) [ER 92].
Estos métodos (con excepción del MCA) se implementaron no sólo en Java (lenguaje elegido para el desarrollo de todas las aplicaciones del proyecto) sino que también se implementaron en C++, de forma de poder hacer también un estudio comparativo de los tiempos de respuesta en los dos lenguajes.

Finalmente incluiremos un resumen de las pruebas realizadas para la validación de la aplicación y el análisis de algunas medidas de interés. Para no sobrecargar este documento, solo se presentarán algunas pruebas de particular interés, incluyéndose en la totalidad de las mismas en la página web del proyecto: http://www.fing.edu.uy/inco/grupos/invop/cda2005.

E.2. Métodos

La presente sección pretende ser una presentación formal de los métodos de evaluación implementados para las diferentes medidas de grafos que se evalúan en la aplicación. Para eso comenzaremos por presentar el modelo de grafos con el que representaremos nuestra realidad y las tres medidas que se pretenden evaluar sobre los mismos. Luego describiremos consecutivamente los tres métodos que se implementaron para la evaluación de estas tres funciones, haciendo una presentación de la teoría de cada uno, como los aplicamos a nuestro problema en particular, y sus ventajas y desventajas.

E.2.1. Modelo

El campo de nuestro estudio son las redes de computadoras, las cuales modelamos mediante grafos no dirigidos, de la forma $G=(V,E)$ en donde V es el conjunto de nodos del grafo y E es el conjunto de aristas de la forma $e_i(v_1, v_2)$ en donde v_1 y v_2 pertenecen a V y son diferentes entre sí.
Asignaremos a cada arista e una probabilidad independiente de funcionamiento r_e, tal que $0 <= r_e <= 1$. De esta forma una red particular está dada por un arreglo $R = [r_{e1}, r_{e2}, \ldots, r_{em}]$ en el que los r_{ei} mayores que 0 indican la presencia de una arista con probabilidad r_{ei} y aquellos $r_{ej} = 0$ indican la ausencia de una conexión directa entre los nodos extremos de la arista e_j.

En particular nuestro modelo esta inspirado por las redes P2P puras. Estas redes se pueden describir como un conjunto de nodos o puertos (peers) interconectados, que tienen la finalidad global de realizar algún tipo de actividad en conjunto y en las que no existe un servidor central que controle o regule el funcionamiento de la red. En las redes consideradas, cualquier nodo puede ser tanto fuente, nodo intermedio o terminal de un pedido particular.
Hemos asumido además que el mecanismo que se emplea para la búsqueda de información es el llamado inundación o broadcast que corresponde a una búsqueda BFS (Breadth First Search) en los nodos: el nodo origen del pedido envía una consulta por la información que necesita a sus vecinos, cada uno de los cuales chequea su base de datos por la información requerida y reenvía el pedido a todos sus vecinos (eventualmente se puede omitir el nodo por el que ingresó el pedido de la lista de nodos de salida), repitiéndose este procedimiento hasta encontrarse el nodo poseedor de la información buscada o hasta alcanzarse un cierto nivel (número de saltos, time-to-live -TTL-, o diámetro máximo -D- desde el nodo inicial de la consulta) de tolerancia predefinido. Llamaremos CCP a la cantidad de conexiones promedio que tiene cada nodo es decir la cantidad de vecinos o nodos que él conoce.

En este contexto la cantidad N de nodos de nuestro grafo representará la cantidad de nodos activos en la red P2P. A su vez este grafo será completo dado que cualquier nodo activo de la red P2P puede virtualmente conectarse con cualquier otro. Por otro lado la cantidad de conexiones que realiza en promedio cada nodo (CCP), se traduce en una probabilidad de utilización de las aristas: si hay N nodos y cada uno realiza CCP conexiones entonces, la probabilidad de que una arista sea utilizada es CCP/(N-1) y este valor es la confiabilidad r_e de las aristas de nuestro modelo.

Siguiendo la motivación expuesta, en las aplicaciones de evaluación y optimización (ver **Apéndice F**) hemos trabajado con grafos completos y aristas equiprobables con la probabilidad de funcionamiento antes mencionada. Esto no quita que la implementación de las estructuras de datos y los algoritmos se hayan hecho de forma genérica, permitiendo la evaluación de grafos con cualquier topología y con cualquier valor de confiabilidad en sus aristas, hecho que se comprueba a través de la integración que se realizó de los algoritmos de evaluación con la herramienta de visualización de grafos, precisamente con el propósito de mostrar la generalidad de las implementaciones realizadas.

Nuestro problema consiste en definitiva en realizar la evaluación del valor esperado de una función Φ que toma como entrada una instancia de un grafo *G* y retorna el valor de una propiedad sobre el mismo.

$$\Rightarrow d = E(\phi(G))$$

E.2.2. Medidas de interés sobre las redes

El problema que se desea solucionar es dado un grafo particular realizar la evaluación de las siguientes medidas:

1. Confiabilidad diámetro acotada de la red.
2. Carga de la red.
3. Porcentaje de nodos conectados.

En lo que sigue presentaremos algunos detalles de cada una de estas medidas.

E.2.2.1. Confiabilidad diámetro acotada.

Una medida clásica de interés sobre redes de comunicación es la confiabilidad de la misma que podemos definir desde dos enfoques diferentes ambos definidos y utilizados en [Maut 00] y [CU 93]:

- Solidez o resistencia a fallos: Medida basada en algunos parámetros de las redes como 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 [PU 96]).

- Confiabilidad de los componentes: 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.

En nuestro caso tomaremos el segundo enfoque, considerándose una medida de confiabilidad basada en la topología de la red y en la confiabilidad de los componentes. En el caso general se asignan probabilidades de funcionamiento a los nodos y aristas de la red, pero en nuestro modelo simplificado los nodos no sufren fallos, por lo que su confiabilidad es 1 simplificando del análisis consecutivo.

En el caso general, la confiabilidad se puede definir como una función de probabilidad con la siguiente forma:

$$\Rightarrow R_K: G \times K \subseteq V \rightarrow [0,1]$$

La evaluación de esta probabilidad $R(G)$, denominada confiabilidad K-terminal, es un problema ampliamente estudiado (por ej. en [CP 01]) y NP-complejo. Esta medida está indicando la probabilidad de que cada par de nodos terminales de *K* esté conectado por un camino operacional, y en la práctica indica la probabilidad de que la red esté funcionando.
A partir de esta formulación general, se definen algunos casos particulares típicos que se estudian ampliamente en la literatura del tema (por ej., en [EKMR 91]):

- $K = V \quad \Rightarrow R_V: G \rightarrow [0,1]$ (confiabilidad todos-terminal)
- $K=\{s,t\} \quad \Rightarrow R_{st}: G \times V \times V \rightarrow [0,1]$ (confiabilidad fuente-terminal)

En nuestro caso consideraremos como medida de confiabilidad la confiabilidad global $R(G) = R_V$, que mediría la probabilidad de existencia de caminos operacionales entre todos los nodos de la red.

Una generalización de este concepto de confiabilidad clásico, es la confiabilidad diámetro acotada ([CP 01] y [CP 04]) que introduce restricciones en el largo (diámetro) de los caminos que conectan pares de nodos en la definición del modelo. En la práctica, esta situación se da cuando tenemos restricciones en el tiempo de vida de los paquetes en la red (TTL), por ej., en los casos en que existen demoras en cada uno de los nodos y el tiempo de comunicación entre un par de nodos debe ser menor a *D* veces esa demora.
En definitiva, la función de confiabilidad diámetro acotada general *R*(*G*, *D*) estaría dada por una función del estilo:

$$\Rightarrow RDA_K: G \times (K \subseteq V) \times [1..N\text{-}1] \rightarrow [0,1]$$

Esta generalización del problema de confiabilidad K-terminal, estaría midiendo la probabilidad de que todo par de nodos terminales en *K* esté comunicado por un camino operacional de largo menor o igual a *D*. En particular, si *G* tiene *N* nodos, *R*(*G*, *N*-1) estaría midiendo la confiabilidad K-terminal clásica *R*(*G*).
En concreto, la medida de confiabilidad que nos interesará evaluar en nuestro estudio será la confiabilidad diámetro acotada para el caso particular *K* = *V*, que queda definida por la función:

$$\Rightarrow RDA_V: G \times [1..N\text{-}1] \rightarrow [0,1]$$

En definitiva, el algoritmo en que nos basaremos para la implementación de la función de CDA consiste en ir "visitando" cada nodo (que denominaremos en el pseudocódigo 'nodoOrigen') y hacer una recorrida BFS en el grafo partiendo de éste, pero encolando solamente los nodos que están a distancia D o menor del mismo. Si terminado el BFS, no están todos los nodos del grafo accesibles, no se continúa con el proceso y se devuelve 0.

```
FUNCTION FiDiametroAcotada (Grafo g; integer D)
     nodoOrigen := 0
     cola := CrearCola()
     Para cada nodo nodoOrigen
          Inicializar(tablaAccesibles)
          AgregarCola(cola, nodoOrigen)
          MarcarAccesible(tablaAccesibles, nodoOrigen)
          Mientras (NOT Vacia(cola))
               (idNodo, distancia) := quitar(cola)
               adyacentes := AdyacentesNoAccedidos(g, idNodo)
               MarcarAccesibles(tablaAccesibles, adyacentes)
               Si (distancia < D)
                    AgregarCola(cola, adyacentes)
               FinSi
          Si (NOT TodosAccesibles(tablaAccesibles)) Retornar 0
     FinPara
     Retornar 1
ENDFUNCTION
```

En el pseudocódigo presentado se utilizan algunas estructuras de datos auxiliares que describiremos a continuación:

- '*g*' representa un grafo y esta dado por dos matrices de NxN, una de ellas de aristas para almacenar las confiabilidades y la otra de booleanos para representar si la arista correspondiente está presente o no en un determinado estado de la red.

- '*cola*' es una cola de elementos compuestos por un nodo v_n y una distancia que representa las distancia a que están los adyacentes de v_n del nodo origen que se está procesando en ese momento.

- '*tablaAccesibles*' es un arreglo de booleanos de tamaño N (la cantidad de nodos del grafo). Si el elemento i de la tabla esta marcado con True, significa que ese nodo i, esta a distancia D o menor del nodo origen que se esté considerando.

E.2.2.2. Carga.

Otra medida importante en nuestro contexto de trabajo es la cantidad de paquetes enviados por los nodos de la red.
Operacionalmente, en una red P2P por ej., esta medida sería un reflejo de la carga promedio que está teniendo la red.

Esta medida depende, al igual que la confiabilidad, de la topología de la red y de las confiabilidades de sus componentes, a lo que se agrega la dependencia de la implementación de la red, y particularmente el algoritmo de distribución de los paquetes en la misma. En nuestro estudio consideraremos un mecanismo de distribución de paquetes básico: la inundación basada en un algoritmo de búsqueda BFS.
En éste, cada nodo al que le llegue un pedido de información (externo al sistema) genera un pedido de la información solicitada que se envía a todos sus vecinos. A su vez, cada uno de estos chequea la existencia de la información requerida en su base de datos y, en caso de no encontrarse la misma, se reenvía el paquete a todos sus vecinos (menos el vecino del que llegó el pedido), continuando la diseminación del pedido de forma recursiva.
Se podría agregar "más inteligencia" al este algoritmo, pero la simplicidad de éste nos permite buscar fórmulas analíticas simples para la cantidad de paquetes circulando en un momento particular, de forma de contar con valores para comparar los resultados que se obtengan mediante la simulación del funcionamiento del sistema.

Una forma sencilla de modelar este algoritmo de diseminación de paquetes, es partiendo de la base de si el número promedio de conexiones permitidas a cada nodo en promedio es *CCP*, y si todas las aristas del grafo son independientes, entonces la probabilidad de que una arista cualquiera independientemente del estado de las demás es *CCP*/*N*, con *N* siendo la cantidad de nodos de la red.
En concreto, cuando un paquete sale de un nodo v_i y llega a otro nodo v_k por la arista v_i-v_k , el nodo v_k reenviará el mensaje a todos sus adyacentes excepto v_i. El nodo v_k tiene *N*-2 posibles vecinos distintos de v_i y cada uno de ellos existe realmente con probabilidad *CCP*/*N*-1. Por lo tanto, el número medio de conexiones que esperamos desde v_k es *CCP*(*N*-1)/*N*.
En definitiva, un mensaje enviado desde un nodo *v*, va a generar C(1+ C(N-1)/N + (C(N-1)/N)^2 + ... + (C(N-1)/N)^(D-1)) mensajes, lo que en se reduce a la siguiente fórmula aproximada para la cantidad de paquetes enviados por los nodos de la red:

$$\Rightarrow C \cong N.CCP\left(\frac{1-CCP^{D}(1-1/N)^{D}}{N-(N-1)CCP}\right)$$

Al igual que para la CDA, presentaremos el algoritmo que seguiremos para la implementación de la evaluación de la función de carga. La estrategia en este caso es "pararse" en cada nodo, contar cuantos paquetes se generan desde él e ir acumulando ese valor. Para saber cuanto paquetes genera cada nodo se hace un BFS, pero solo encolando los nodos que están a distancia menor que D y además sin ir marcando los visitados pues un nodo puede estar accesibles desde el nodo origen a través de varios caminos. Finalmente se multiplica el resultado por CPN, la cantidad de pedidos promedio que realiza cada nodo en una unidad de tiempo.

```
FUNCTION FiCarga (grafo g; integer D, double CPN)
     resultado := 0;
     Para cada nodo nodoOrigen
          nroPaquetes := -1
          AgregarCola(cola, nodoOrigen, nodoOrigen, 0)
          Mientras (NOT Vacia(cola))
               (idNodo, idOrigen, prof) := Quitar(cola)
               nroPaquetes := nroPaquetes + 1
               adyacentes := Adyacentes(g, idNodo, idOrigen)
```

```
                profAct := prof + 1
                si (profAct < D)
                     AgregarCola(cola,    adyacentes,    idNodo,
profAct)
                si no
                     nroPaquetes := nroPaquetes + cantAdyacentes
                fin si
           FinMientras
           resultado := CPN(resultado + nroPaquetes)
      FinPara
FINFUNCTION
```

La estructura de datos auxiliar que se utiliza en este caso es '*cola*', la cual es una cola cuyos elementos están compuestos por el nodo actual, el nodo por donde llegue al actual (para no tomarlo en cuanta cuando se pide los adyacentes de nodo actual) y la distancia a la que se encuentra el nodo actual del nodo origen que se está procesando en ese momento.

El método *Adyacentes*(*g*, *nodoA*, *nodoB*) retorna los adyacentes de *nodoA*, en el estado actual de *g*, sin considerar a *nodoB* en ese conjunto de adyacentes.

E.2.2.3. Porcentaje de nodos conectados.

Para nuestra definición de la función de confiabilidad diámetro acotada con que trabajaremos consideramos la medida de probabilidad de que todos los nodos, o sea que $K = V$, estén comunicados entre sí por caminos de largo a lo sumo *D*.
Si ahora consideramos $K = \{s, t\}$, donde *s* y *t* son nodos del grafo *G*, podemos definir de forma análoga una función de confiabilidad RDA_{st} que indique la probabilidad de que el par de nodos *s* y *t* esté conectado por un camino de largo menor o igual a *D*.

$$\Rightarrow RDA_{st}: G \times \{s, t\} \times [1..N\text{-}1] \rightarrow [0,1]$$

La función RDA_{st} expresa la probabilidad de que un nodo específico de *G* pueda comunicarse por un camino de largo máximo *D* con otro nodo del grafo. Si miramos entonces el promedio de todos los valores de RDA_{st} para todas las combinaciones posibles de nodos *s* y *t* pertenecientes al conjunto de nodos del grafo, este valor es equivalente al porcentaje de nodos conectados por caminos de estas características.

Esta medida es de interés principalmente para grafos en los que su confiabilidad diámetro acotada sea menor que 1, situación ésta que es el común de lo casos para grafos medianos y grandes.
Dado que en nuestro proyecto el interés está dado en el estudio de valores globales de la red, y no en valores propios de los nodos, tomaremos la función P, que se define a continuación, para el estudio de la conectividad de las redes y para la posterior implementación en las aplicaciones, en lugar de estudiar la función RDA_{st} antes detallada.
La función de porcentaje de nodos conectados por caminos operacionales diámetro-confiables, se basa entonces en la obtención de todos los pares de nodos conectados por caminos de diámetro menor o igual a *D*, y en su división por el total de pares de nodos del grafo calculado en base a la cantidad de nodos del mismo.

$$\Rightarrow P: G \times [1..N] \rightarrow [0..100]$$

$$\Rightarrow \mathrm{P(G, D)} = \frac{\left[\text{Cantidad de pares de nodos conectados por caminos de largo} \le \mathrm{D}\right]}{N * (N-1)/2}$$

El algoritmo para la función de porcentaje de nodos conectados por caminos de largo menor o igual que *D* sigue la misma idea que la presentada para la función de CDA, con la diferencia que al final de cada BFS lo que se hace es contar cuantos nodos en la tabla '*tabalAccesibles*' están marcados y se va acumulando ese número variando el nodo origen de la búsqueda en el conjunto de nodos del grafo. Finalmente se devuelve que porcentaje del total de pares nodos

del grafo (N(N-1)/2, como se ve en la fórmula de arriba) es esa cantidad acumulada dividida entre dos, porque cada camino se contabilizó dos veces. Las estructuras de datos auxiliares que se utilizan son las mismas que las descritas junto al pseudocódigo de la función de evaluación de la CDA.

```
FUNCTION FiPorcentaje (grafo g; integer D)
     nodoOrigen := 0
     chance := True
     cola := crearCola()
     Para cada nodo nodoOrigen
          Inicializar(tablaAccesibles)
          AgregarCola(cola, nodoOrigen)
          MarcarAccesible(tablaAccesibles, nodoOrigen)
          Mientras (NOT Vacia(cola))
               (idNodo, distancia) := quitar(cola)
               adyacentes := AdyacentesNoAccedidos(g, idNodo)
               MarcarAccesibles(tablaAccesibles, adyacentes)
               Si (distancia < D)
                    AgregarCola(cola, adyacentes)
               FinSi
          paresNodos                                          :=
paresNodos+CantAccesibles(tablaAccesibles)
     FinPara
     retornar 100*(paresNodos/2)/(N*(N-1)/2)
FINFUNCTION
```

E.2.3. Métodos

E.2.3.1. Generación Completa de Estados o GCE

Un posible encare para la evaluación de las tres medidas de interés, es generar todos los estados posibles del grafo y obtener el valor de Φ para cada uno de ellos. Luego se acumulan las medidas obtenidas para los $2^{\#(E)}$ estados posibles, ponderando cada una por la probabilidad de ocurrencia del grafo a partir del cual se obtuvo.
Un pseudocódigo para la GCE se presenta a continuación:

```
FUNCTION GeneracionCompletaEstados
    R := 0
    Gi := EstadoInicial()
    While (NOT UltimoEstado(Gi))
         p := ProbabilidadOcurrencia(Gi)
         R := R + (Φ(Gi) * p)
         Gi := SiguienteEstado(Gi)
    EndWhile
ENDFUNCTION
```

E.2.3.2. Monte Carlo crudo o MCC

Dado que la cantidad de estados a evaluar usando GCE crece exponencialmente con el número de aristas del grafo en consideración (que en nuestro caso es *N* * (*N* - 1) / 2, dado que trabajamos con grafos completos), éste método se vuelve rápidamente ineficiente al crecer el tamaño del grafo.
Una de las fórmulas más difundidas de realizar una estimación del valor de la evaluación de una Φ, consiste en tomar una serie al azar de estados del grafo del espacio de estados, y estimar el valor de la medida *R*, usando un estimador *R** definido en función de esta serie de resultados obtenidos de la siguiente forma:

$$\Rightarrow R^* = \frac{1}{M}\sum_{i=1}^{M}\phi(G_i)$$

en donde los G_i son realizaciones aleatorias independientes del estado del grafo que modela la red, donde cada arista del mismo está presente o no según el modelo de probabilidad descrito anteriormente. En este contexto *M* es la cantidad de estados del grafo que se consideran para la evaluación particular.

Dado que esta fórmula es un estimador del valor real de la función, resulta de interés determinar la calidad de este estimador a través de la obtención de su varianza y, a partir de esta, los intervalos de confianza.
La varianza para el estimador que definimos -*R**- sería entonces:

$$\Rightarrow Var(R^*) = \frac{\sum_{i=1}^{M}\phi(G_i)^2}{M(M-1)} - \frac{R^{*2}}{M-1} = V$$

Un pseudocódigo para el MCC a implementar sería como sigue:

```
FUNCTION MonteCarloCrudo
    R := 0
    V := 0
    For n := 1 To N
          SortearEstado(Gi)
          R := R + Φ(Gi)
          V := V + Φ(Gi)²
    EndFor
    R:= R / M
    V:= V / (M (M - 1) - V²(M - 1))
ENDFUNCTION
```

E.2.3.3. Monte Carlo Antitético generalizado o MCA

El MCC, a pesar de ser de ser de gran utilidad al permitir la evaluación de las funciones en un mayor número de casos que la GCE, tiene una limitante importante en cuanto a la calidad de los resultados que se obtienen al acercarse a 1 la confiabilidad de los grafos que se consideran.
Se han estudiado diferentes técnicas en pos de una reducción de la varianza de los resultados que se obtienen, de las que podemos mencionar la Reducción de Varianza Recursiva ([Maut 00] y [CEK 94]). Por un repaso del estado del arte sobre técnicas de reducción de varianza sobre el MCC, dirigirse a [CU 94].

El método que utilizaremos en este proyecto será el desarrollado en [EKR 92] basado en el uso de variables antitéticas, que tiene beneficios tanto en tiempo de ejecución como en la precisión de las soluciones que se obtienen en comparación con el MCA.

Para presentar el método comenzaremos viendo el caso para bloques de tamaño *L* = 2 que tomaremos de base para presentar el caso general que en definitiva será el que implementaremos y del cual presentaremos en este documento un pseudocódigo simplificado.
La implementación del método de Monte Carlo que presentamos arriba puede ser generalizada de la siguiente forma:

```
FUNCTION MonteCarloAntitetico_Base
    R := 0
    V := 0
```

```
        For b := 1 To B
            For e := 1 To E
                U := VariableUniforme(0, 1)
                X[1][e] := U < r[e]
                X[2][e] := 1-U < r[e]
            EndFor
            Rb := (Φ(X[1][e]) + Φ(X[2][e]))/2
            R := R + Rb
            V := V + Rb^2
        EndFor
        R:= R / B
        V:= V / (B(B - 1) - R^2(B - 1))
    ENDFUNCTION
```

Este método se basa en la propiedad de las variables aleatorias uniformes (representadas en el ejemplo por `U`) de que su complemento (1 - `U`) también es una variable aleatoria uniforme, haciendo posible generar dos estados de cada arista por cada sorteo de la misma. De esta forma con la misma cantidad de sorteos estamos sorteando el doble de estados del grafo que en el MCC que planteamos.
Desarrollando un poco la notación presentada en el pseudocódigo, los arreglos `X[i][e]` representan los estados del grafo que estamos generando; `Rb` representa el estimador *R** presentado más arriba para el caso particular *N* = 2; mientras que el vector `r[e]` almacena las confiabilidades asociadas a cada una de las aristas del grafo.

El hecho de relevancia aquí es que la covarianza entre las variables aleatorias X_e^1 y X_e^2 es negativa y si la Φ es una función monótona creciente de sus argumentos, también se cumple que Cov($\phi(X_e^1), \phi(X_e^2)$) < 0, por lo que esta técnica produce una menor varianza que el MCC (por un desarrollo completo y demostración de este enunciado, dirigirse a [EKR 92])

Una manera diferente de interpretar el caso base presentado arriba podría ser como sigue:

```
FUNCTION MonteCarloAntitetico_Base2
    R := 0
    V := 0
    For b := 1 To B
        For e := 1 To E
            X[1][e] := X[2][e] := Xdef[e]
            U := VariableUniforme(0, 1)
            Case
                0 <= U < q[e]: X[1][e] := 1 - Xdef[e]
                q[e] <= U < 2*q[e]: X[2][e] := 1 - Xdef[e]
            EndCase
        EndFor
        Rb := (Φ(X[1][]) + Φ(X[2][]))/2
        R := R + Rb
        V := V + Rb^2
    EndFor
    R:= R / B
    V:= V / (B(B - 1) - R^2(B - 1))
ENDFUNCTION
```

Algunas variables usadas en este psudocódigo son las que siguen:

- `Xdef[e]`: Define un arreglo de booleanos que contiene el estado más probable de cada una de las aristas del grafo.
- `q[e]`: Vector con las probabilidades de que cada una de las aristas se encuentre en su estado menos probable.

`q[e]` induce para cada arista una partición en los intervalos I_e^1 = [0, q_e[, I_e^2 = [q_e, 2q_e[, y J_e = [2q_e, 1], de forma que en caso de la variable `U` esté dentro de uno de los intervalos I_e^i será cambiado el estado de la arista *e* en X_e^i.

Generalizando este razonamiento, podemos escribir a las variables aleatorias X_e^i de la siguiente manera:

$$\Rightarrow X_e^i = (1 - Xdef[e])1_{(U \in I_e^i)} + (Xdef[e])1_{(U \notin I_e^i)} \qquad i = 1,...,s_e$$

en donde s_e representa la cantidad de estados que se generan para X_e, siendo s_e = 2 en el caso presentado arriba.

Generalizando estas ideas para un valor de *L* mayor que 2, podemos desarrollar el siguiente pseudocódigo:

```
FUNCTION MonteCarloAntitetico
     Preliminares()

     R := 0
     V := 0
     For b := 1 To B
          InicializarBloque()
          EstadosModificados()
          For l := 1 To L
               Rb := Rb + Φ(X[l][])
          EndFor
          Rb := Rb / L

          R := R + Rb
          V := V + Rb^2
     EndFor
     R:= R / B
     V:= V / (B(B - 1) - R^2(B - 1))
ENDFUNCTION
```

Comparando con respecto a la segunda formulación que se presentó para el caso base, este pseudocódigo modulariza la inicialización y carga de las posiciones de la matriz `X[][]` para cada arista *e*, y además se generaliza la suma de las diferentes evaluaciones para cada uno de los *L* estados sorteados en cada iteración.

Aclarando este algoritmo presentado, *L* define entonces el tamaño de los bloques considerados y será un parámetro de entrada del método, mientras que *B*, como antes, define el tamaño de cada uno de estos bloques.
Para cada bloque se comienza por inicializar la matriz de estados *X*, seteando las *L* filas de la misma con el vector *Xdef* que contiene el estado más probable de cada uno de las *E* aristas del grafo. Esto se realiza en el llamado a la función `InicializarBloque()`.

A continuación, en el llamado a la función, `EstadosModificados()`, procedemos a efectuar los sorteos necesarios para la determinación de cada uno de los *L* estados que produciremos y posteriormente evaluaremos. Para ello, necesitamos hacer algunas definiciones previas.
Dado que el intervalo J_e = [2q_e, 1] puede ser mayor que q_e, podemos definir un tercer intervalo I_e^3 = [2q_e, 3q_e[. Este razonamiento puede ser repetido hasta que el tamaño del intervalo J_e sea menor que q_e, lo que se va a dar luego de a lo sumo $\lfloor 1/q_e \rfloor$ divisiones de J_e. De esta forma

podemos elegir cualquier s_e de forma que cumpa que 1 <= s_e <= $\lfloor 1/q_e \rfloor$. Para una aplicación óptima de la mejora inducida por el uso de variables antitéticas, se aconseja tomar s_e=$\lfloor 1/q_e \rfloor$. [Fish 86].
De esta forma podemos dividir los *L* estados que generaremos para cada arista *e*, en nbs_e subbloques de tamaño s_e, en donde nbs_e= *L* / s_e.
Para generar entonces los *L* estados del grafo de cada iteración, procederemos a realizar un sorteo de una variable aleatoria uniforme *U*, por cada subbloque de cada arista *e* definida, y según el intervalo I_e^h en donde caiga *U* se modificará el estado de la posición de la matriz que corresponda.

Para todo este proceso dentro de la función `EstadosModificados()` son necesarias una serie de estructuras auxiliares para el almacenamiento de estos valores que mencionamos arriba. Esto se realiza en el procedimiento `Preliminares()` que se ejecuta por única vez al comienzo. En el mismo, partiendo de la base de que se tiene un vector `r[e]` con las confiabilidades de cada una de las aristas del grafo, para cada arista *e* se calculan los siguientes valores:

- `q[e]`: Probabilidad de que la arista esté en su estado menos probable.
- `s[e]`: Tamaño de cada subbloque.
- `nbs[e]`: Cantidad de subbloques por bloque.
- `Xdef[e]`: estado más probable de la arista.

Algunas mejoras de eficiencia que se podrían hacer sobre este algoritmo serían la preevaluación del valor de $\phi(Xdef)$ y tener en cuenta (para los casos en que la función a evaluar corresponda a una función de confiabilidad del grafo) que la red no puede estar caída si el número de conexiones caídas es menor que el tamaño de un corte minimo del subgrafo inducido por el conjunto de aristas activas.
En nuestra implementación solo tendremos en cuenta la primera de estas consideraciones, que puede ser importante en la medida de que las confiabilidades de las aristas sean grandes y la cantidad de aristas no sea muy elevada. Por detalles de las ganancias logradas por la implementación de estas optimizaciones sobre el algoritmo, dirigirse nuevamente a [EKR 92].
También se pueden hacer optimizaciones en cuanto al requerimiento de memoria del algoritmo, que puede ser de consideración al crecer el tamaño de *L*. Para esto se propone usar una forma de almacenamiento compacto para la matriz `X[][]`, lo cual se implementa mediante la utilización de dos vectores auxiliares y algunas variables escalares. En lugar de construir "físicamente" la matriz, la idea es mantener una tabla virtual en la en la que se indique para cada arista *e*, dónde está el primer elemento diferente de `Xdef[e]`.

E.3. Objetivos y requerimientos de la aplicación

E.3.1. Objetivos

Si miráramos la aplicación como un desarrollo aislado, sus objetivos no serían más que implementar de manera los más eficiente posible (básicamente en cuanto a tiempo de ejecución y utilización de memoria) el cálculo de 3 propiedades de un grafo, a saber: la confiabilidad diámetro acotada, el porcentaje de pares de nodos conectados a distancia menor que un *D* dado, y la cantidad de paquetes promedio circulando por la red.

Este objetivo básico se mantiene al pasar a analizar el entorno en que se desarrollará la aplicación en nuestro caso en particular. Sin embargo a partir de algunos factores definidos por este entorno, podemos refinar otros objetivos.
Dado que la idea de la aplicación es interactuar con otras aplicaciones (en principio con el Visualizador de Grafos y el Optimizador, pero dejamos abierta la posibilidad de que sea cualquier otra) su motor de cálculo o capa lógica, deberá ser lo más flexible posible como para recibir entradas (grafos) de diferentes formas. El hecho de que los algoritmos de evaluación, o por lo menos alguno de ellos, van a ser parte del Optimizador es otro hecho que aumenta el peso del objetivo de maximizar la eficiencia del procesamiento.
Este objetivo se alinea también con el hecho de que nuestro proyecto forma parte de un proyecto macro, con el que se pretende también interactuar, lo que requiere interfaces bien definidas y estructuras de datos flexibles para una fácil adaptación para la interacción con otras aplicaciones.

No por estar incluido dentro de los objetivos de todos los desarrollos que encaramos en este proyecto, dejaremos de mencionar en esta oportunidad el desarrollo de la aplicación pensado en la mantenibilidad y extensibilidad de la misma.

En lo que sigue de esta parte del documento se desarrollarán consecutivamente: los requerimientos planteados para el desarrollo; el diseño de la aplicación mediante Diagramas de Clases y Diagramas de Colaboración; finalizando con conclusiones sobre el desarrollo realizado y trabajos futuros asociados al mismo.

E.3.2. Requerimientos

Entrando en lo que son los requerimientos para la aplicación y tomando como base los objetivos que se acaban de mencionar, podemos armar la siguiente lista:

- Desarrollo de una aplicación para la evaluación de tres funciones de interés sobre grafos completos:
    - Probabilidad de que todos los nodos puedan alcanzar al resto de los nodos a través de caminos de largo menor o igual a *D*.
    - Porcentaje de pares de nodos comunicados por caminos de diámetro a lo sumo igual a *D*.
    - Carga de paquetes enviados por los nodos de la red, determinada a partir de la cantidad de pedidos de información que llegan a cada nodo en promedio.

  La implementación de estas funciones requiere la alimentación de las mismas por un conjunto de parámetros que determinarán además el formato del grafo sobre el que se trabajará. Estos parámetros son:
    - La cantidad de nodos de la red → *N*
    - La cantidad de conexiones permitidas a cada nodo en promedio → *CCP*
    - La cantidad de pedidos promedio que llegan a cada nodo → *CPN*
    - El largo máximo para que un camino entre un par de nodos sea considerado como operacional → *D*

- Implementación en Java de tres métodos diferentes para el cálculo de estas tres funciones mencionadas.

- Implementación en C++ de al menos uno de los tres métodos implementados en Java, e integración de la ejecución del mismo a la interfaz de la aplicación.

- Posibilidad de selección sencilla de las funciones a evaluar y de los métodos a través de los cuales se realizará la evaluación

E.4. Diseño

E.4.1. Consideraciones generales de diseño

En esta sección describiremos algunos aspectos esenciales del diseño de la aplicación y comentaremos las decisiones de diseño que se tomaron y sus por qué. Todo esto se complementará mediante los correspondientes diagramas de colaboración para los algoritmos más importantes, y el diagrama de clases de la aplicación.

Para empezar, y siguiendo todo lo mencionado en la sección de Objetivos, se buscó un diseño sencillo y flexible tanto para medir las propiedades a evaluar en el grafo, como para los diferentes algoritmos con que se implementarán las mismas. Que más sencillo y flexible que utilizar la abstracción y definir clases abstractas con la interfaz de operaciones a implementar por cada una de las variantes. Esto fue precisamente lo que se hizo en estos casos.
Las propiedades que se van a evaluar en cada instancia particular del grafo (las Φ's) extienden a una clase abstracta que contiene la operación de evaluación y que cada una está obligada a implementar. De esta forma, si se pretendiera agregar una nueva Φ, inicialmente bastaría con que la clase que implementa a la misma extienda a la clase abstracta e implemente la operación de evaluar.
De manera totalmente análoga se creo una clase abstracta para que oficie de superclase de todos los métodos a través de los cuales se realiza la evaluación del grafo genérico que se recibe como entrada de la aplicación.

La aplicación cuenta con dos capas, presentación y lógica, que buscamos hacer independientes mediante la utilización de interfaces mutuas entre las mismas, de forma de que cada una no necesite conocer la implementación de la otra, siempre y cuando ambas se plieguen a lo que son las interfaces definidas.
La necesidad de esta comunicación en ambas direcciones se debió a la decisión que se tomó de incluir una barra de progreso en la ventana de la aplicación. Esto generó la necesidad de que información (específicamente, información referente a que método se está ejecutando y al grado de avance en la ejecución del mismo) de la capa lógica pasara a la capa de presentación de forma “continua”.
En definitiva, el mantenimiento de la independencia de capas en este escenario de comunicación bidireccional entre las mismas se resolvió utilizando las siguientes clases:

- *FactoryThreadCalculo*: Oficia de fábrica de objetos *IThreadCalculo* los que en definitiva contienen la interfaz con que la capa de presentación podrá acceder a la ejecución de los diferentes métodos de cálculo provistos por la lógica. En nuestro caso particular esta interfaz es implementada por la clase *ThreadCalculo* las cuales solo pueden ser creadas por ésta fábrica (que además es Singleton, o sea que solo puede existir una instancia de la misma en el sistema)
- *IFramePrincipal*: Interfaz para el acceso a las operaciones de la capa de presentación que interesan a la lógica, las que, como se mencionó arriba, solo son las encargadas de actualizar la barra de progreso más las encargadas de desplegar los resultados retornados por los métodos de cálculo.

Otra consecuencia interesante producto de la decisión de agregar una barra de progreso en la ventana de la aplicación, es la aparición de un nuevo hilo de ejecución (además del hilo principal) en el que se realizarán todo el proceso de cálculo. El mismo está implementado en la clase *ThreadCalculo* que implementa la interfaz *java.lang.Runnable* con este propósito.
La aparición de esta separación en hilos de todo lo que es la parte gráfica de la aplicación de lo que es la parte de cálculo, permite que el usuario tenga a disposición prácticamente en tiempo real, la información de lo que está pasando durante el procesamiento.
El uso de hilos permitió además implementar correctamente la funcionalidad de cancelar el procesamiento (mediante el botón homónimo en la ventana de la aplicación), evento que puede ser capturado y procesado por el hilo principal (en el que queda ejecutando la parte de

presentación de la aplicación) sin necesidad de esperar que culmine el procesamiento de la parte lógica, que es lo que sucedería si se ejecutara todo bajo un mismo hilo. Esta funcionalidad es muy importante en este tipo de aplicación en el que la ejecución con algunos conjuntos de parámetros puede implicar un tiempo de cálculo elevado y una ocupación de los recursos del sistema que puede llegar a bloquear el funcionamiento de la máquina, todo lo cual se soluciona teniendo esta posibilidad de cancelar en cualquier momento el procesamiento.

Es importante destacar otro punto con respecto a las dos capas de esta aplicación. Los métodos implementados, que se detallarán en una sección subsiguiente, fueron diseñados para recibir como parámetro de entrada lo que nosotros denominamos un grafo genérico.
El mismo consiste formalmente en un grafo no dirigido G = (V, E) formado por un conjunto de nodos V y un conjunto de aristas E que conectan los nodos. Este grafo representaría una red de comunicaciones en donde V sería el conjunto de terminales de la red y E representaría el conjunto de líneas de comunicación que unen estas terminales.
Según nuestro modelo, las terminales tienen probabilidad de falla 0 por lo que en nuestro grafo el único atributo que tendrán los nodos será un identificador unívoco en el grafo. Por otro lado, cada línea de comunicación c en E tiene asociada una probabilidad de falla dada por q_c, lo cual se verá reflejado en nuestro grafo en un atributo de las aristas que representará la probabilidad de que la misma esté activa en un momento determinado (o sea que este atributo de las aristas sería igual a 1- q_c)
Esta forma de construcción de las clases que implementan los métodos de cálculo, permite tener diferentes opciones para la construcción de este grafo genérico a nivel de la capa de presentación o de alguna clase intermedia entre la gráfica y la lógica dentro de ésta última. Algunas opciones para la construcción de este grafo genérico podrían ser:

- A partir de un conjunto de parámetros numéricos obtener los valores de confiabilidad de las aristas.
- Permitir al usuario construir directamente o levantar de un archivo, un grafo con las confiabilidades de las aristas.

Para el caso particular de la aplicación de Evaluación, la opción elegida es la primera: el usuario a través de la interfaz gráfica de la aplicación ingresará, entre otros parámetros, la cantidad de nodos (N), la cantidad de conexiones promedio por nodo (CCP), la distancia máxima que se permitirá para los caminos en el grafo (D) y la cantidad de pedidos por nodo (CPN). Específicamente para la construcción del grafo, se utilizarán N y CCP, definiéndose la confiabilidad de todas las aristas del grafo CCP/N-1.
Este diseño fue realizado teniendo en mente que toda la lógica de evaluación de las funciones objetivo en el grafo genérico no solo será utilizada en esta aplicación, sino que también se utilizarán en la aplicación de Optimización y en el Visualizador de Grafos. Para la integración con esta última aplicación es de vital importancia la posibilidad de pasar al motor de cálculo un grafo que viene armado como tal y no viene dado implícitamente por un conjunto de parámetros.

Cumpliendo con uno de los requerimientos planteados para la aplicación, los algoritmos involucrados en la evaluación de las funciones definidas fueron implementados tanto en Java como en C++ (salvo el caso del MCA que solo se implementó en Java). Para facilitar al usuario la elección del lenguaje a utilizar, se decidió unificar el acceso a los algoritmos en ambos lenguajes a través de una misma aplicación.
Dado que esta aplicación, al igual que el resto de las aplicaciones del proyecto, es desarrollada en Java, se definió e implementó una interfaz con las operaciones implementadas en C++ mediante JNI (Java Native Interface) de forma de enmascarar bajo llamado a procedimientos Java el acceso a los algoritmos implementados en C++.

En las siguientes sub-secciones se desarrollarán consecutivamente los diseños de las funciones objetivos sobre una instancia particular del grafo y los diseños de las meta heurísticas a implementar para la evaluación de estas funciones sobre el grafo genérico.

E.4.2. Diseño de las funciones de evaluación

En esta sección mostraremos, mediante diagramas de colaboración, la implementación de las tres funciones de evaluación sobre el grafo que se desarrollaron.

E.4.2.3. Confiabilidad Diámetro Acotada

Dejamos para el final la medida que tal vez presente una innovación más importante en lo que respecta a nuestro proyecto. La función de Confiabilidad Diámetro Acotada retorna un booleano que será True si y solo si cualquier nodo puede acceder a información de otro nodo del grafo a través de un camino de diámetro menor que D, donde D (al igual que la CPN y la instancia particular del grafo) es un parámetro de entrada de la función.
El diseño que realizamos es prácticamente el mismo que para la medida de porcentaje: se divide el problema en N sub-problemas en los que en cada uno de ellos se intenta determinar la posibilidad de llegar al resto de los nodos del grafo a través de caminos de diámetro menor o igual que D. La resolución de cada uno de estos sub-problemas, como se verá en el diagrama de colaboración a continuación, también es muy similar a la que se utilizó para la función de porcentaje. La diferencia fundamental es que en este caso, en lugar de contar la cantidad de nodos a los que llegamos a través de caminos que cumplen la restricción planteada, evaluamos la posibilidad de llegar desde el nodo base del sub-problema a el resto de los nodos del grafo, y en caso negativo se termina el procesamiento retornándose False.

El siguiente diagrama de colaboración muestra el diseño elegido para esta operación:

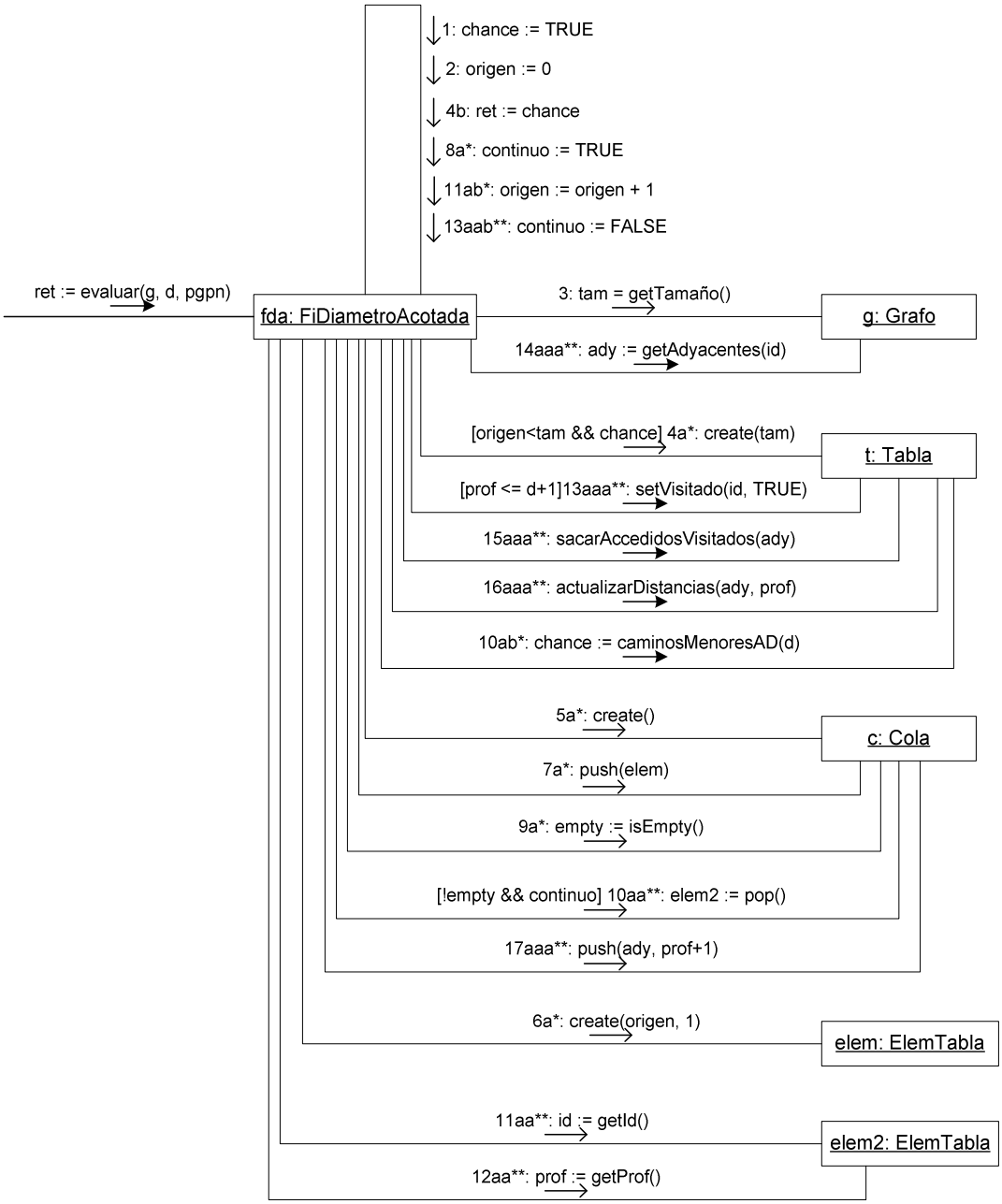


Fig. E.1: Diagrama de colaboración – FiDiametroAcotada:evaluar()

E.4.2.2. Porcentaje de pares de nodos conectados

Esta función de evaluación toma, al igual que el resto, como entradas una instancia particular del grafo genérico que se está evaluando, un límite para el diámetro de los caminos válidos en nuestra realidad (que denominaremos por D), y la cantidad promedio de pedidos que llegan a cada nodo del grafo (que denominaremos CPN). El cálculo que se realiza a partir de estos datos es el de la cantidad de pares de nodos que se encuentran conectados por un camino de diámetro menor o igual a D.

Para la implementación de esta medida se dividió el problema del cálculo global de la cantidad de pares de nodos conectados, en N (donde N es la cantidad de nodos del grafo) problemas de calcular la cantidad de nodos conectados por un camino de distancia menor o igual que D a un nodo particular N_i del grafo.

Para cada uno de estos N sub-problemas serán utilizadas algunas estructuras de datos auxiliares que pasaremos a detallar a continuación:

- Una cola en la que iremos agregando los nodos adyacentes a los nodos que vamos visitando, y de la que iremos sacando los nodos que recorremos en cada paso.
- Un vector en el que almacenaremos los nodos adyacentes al nodo que estamos visitando.
- Una tabla con la información de los nodos que se determinó son accesibles a distancia a lo sumo D desde el nodo origen del subproblema.

Teniendo esto en mente, la implementación que realizamos para esta medida es como sigue: para cada nodo v_B (que denominaremos nodo base del sub-problema) obtenemos sus nodos adyacentes (aquellos conectados por una arista con v_B) y, controlando que la distancia de estos nodos al nodo base sea menor que D, se encolan en la cola. Posteriormente se marcan todos estos nodos como accedidos (o sea, se puede llegar desde N hasta ellos por un camino de diámetro menor o igual que D); se saca el primer nodo de la cola y se repite todo el proceso que acabamos de comentar. Este procesamiento recursivo termina cuando vaciamos la cola, cosa que sucede de forma segura ya que la inserción de nodos en la misma se detiene cuando llegamos a una distancia mayor que D desde el nodo base.

Todo este proceso se muestra en detalle en el siguiente diagrama de colaboración para la función de Porcentaje:

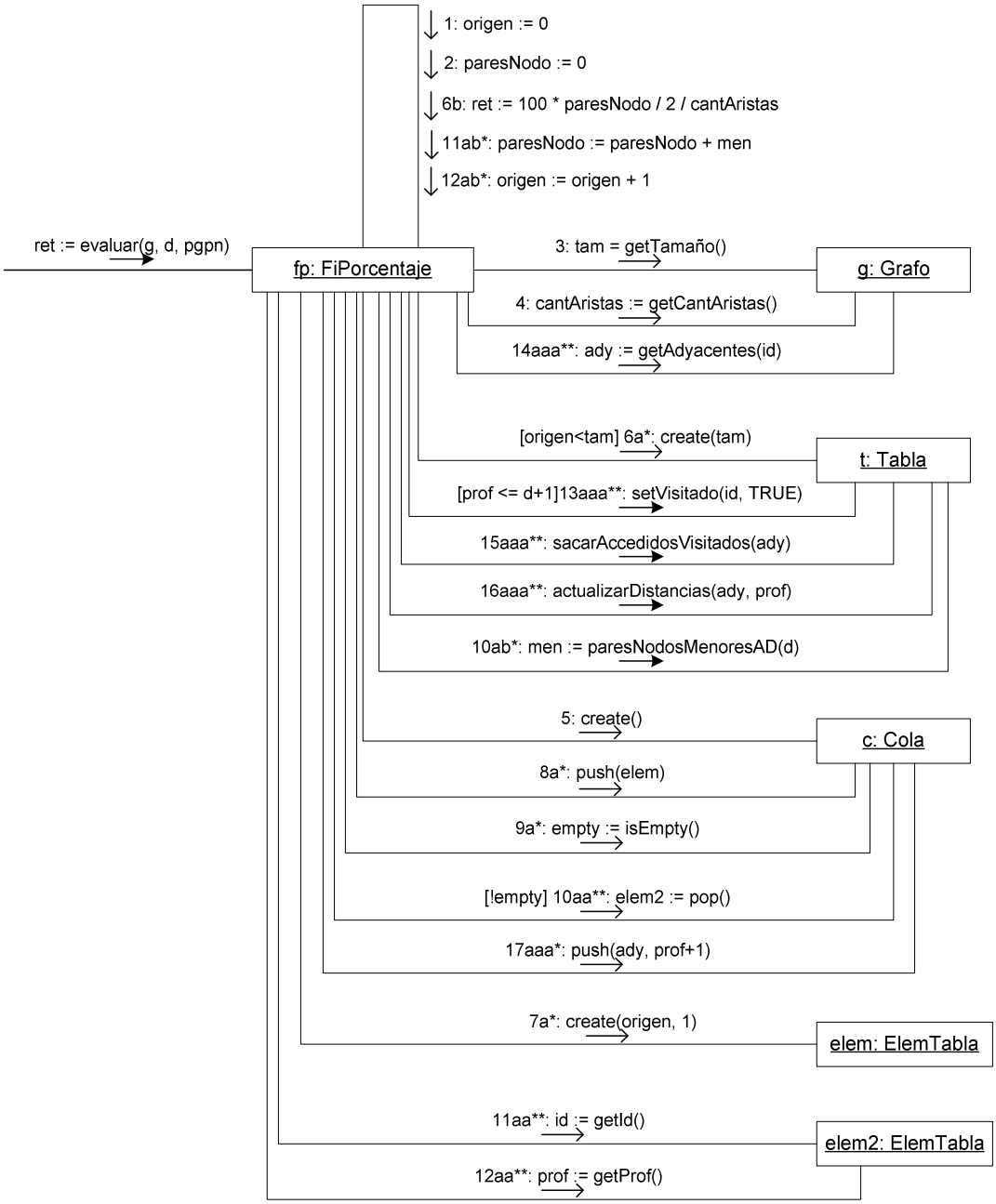


Fig. E.2: Diagrama de colaboración – FiPorcentaje:evaluar()

El resultado final del cálculo surge de la suma de los nodos que se encuentran a distancia menor o igual que D del nodo base, para cada uno de los sub-problemas que mencionamos. Esta suma se divide entre 2, obteniéndose S, ya que cada par de nodos (N_o, N_d) que cumplen la restricción es contado dos veces: una cuando se resuelve el sub-problema con nodo base N_o, y otro cuando se resuelve el sub-problema con nodo base N_d. El porcentaje de nodos conectados por caminos de largo menor o igual a D es entonces el valor S.

E.4.2.1. Carga

La primera función que presentaremos será la que denominaremos función de carga, la cual se encarga de calcular la cantidad de paquetes enviados por los nodos en una instancia determinada del grafo de entrada.

Con la cantidad de paquetes circulando nos referimos a la cantidad de envíos generados en todos los nodos del grafo, a partir del conjunto de pedidos que ingresan al sistema. Para el caso particular de un nodo i, un pedido particular P genera el envío de V_i paquetes generados en i, donde V es la cantidad de vecinos de i. Para este mismo paquete P se generarán V_{Ai} envíos desde cada uno de los A_i nodos adyacentes a i, para cada uno de los cuales se puede aplicar este mismo razonamiento recursivamente hasta alcanzar una distancia D-1 con respecto al nodo original i.
Este mismo algoritmo se aplica para cada uno de los nodos de la instancia particular del grafo que estemos considerando, para obtener la cantidad de paquetes circulando en el grafo.

A continuación se presenta el diagrama de colaboración correspondiente a esta Fi:

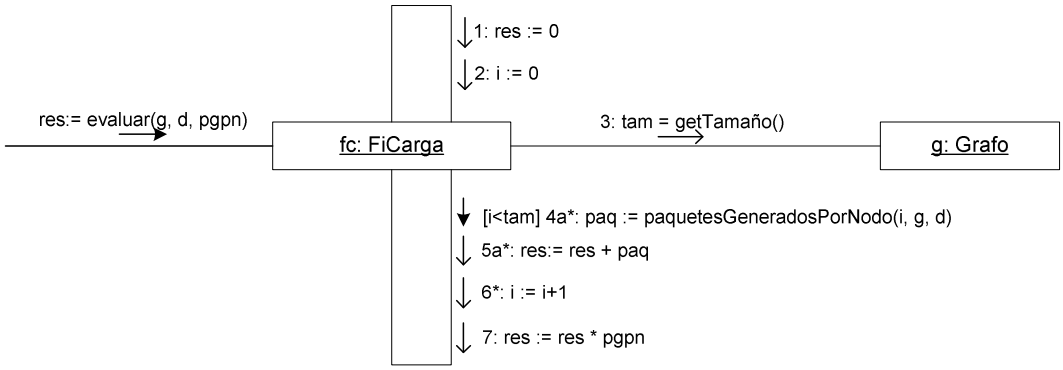


Fig. E.3: Diagrama de colaboración – FiCarga:evaluar()

La interpretación de este diagrama es bastante intuitiva: para cada nodo del grafo se calculan la cantidad de paquetes generados por el mismo a partir de un único pedido, los cuales se van acumulando y al final se multiplican por los Paquetes Generados Promedio por Nodo, valor este que estima la cantidad promedio de pedidos que llegan a cada nodo del grafo.

E.4.3. Diseño de los métodos

Como se mencionó en las consideraciones generales de esta sección, la entrada para todo lo que es el motor de evaluación de las funciones objetivo es lo que dimos en llamar un grafo genérico. Dado un conjunto de funciones objetivo que se pretenden evaluar sobe el mismo diferentes algoritmos que se podrían utilizar para la realización de ésta tarea de evaluación. Las tres que elegimos en nuestro caso, se basan en lo mismo: ir generando diferentes instancias del grafo genérico, realizando la evaluación de las funciones sobre las mismas, y finalmente realizando algún promedio de los resultados obtenidos. Lo que diferencia a cada una de ellas es la elección de las instancias del grafo que se evaluarán y la forma de realizar el promedio de los resultados obtenidos. Los detalles de cada una en estos aspectos se irán desarrollando en los puntos correspondientes a cada una en la presente sub-sección.

Estas instancias del grafo genérico que vamos evaluando, se generan de éste a partir de una serie de elecciones (uno por cada arista) en las que se determina la presencia o no de cada arista en la instancia particular, de forma que para todas las aristas se cumpla que la cantidad media de apariciones dentro del conjunto de instancias generadas sea lo más cercana posible a la probabilidad de ocurrencia que traía la misma en el grafo genérico de entrada.

Los resultados exactos de nuestra evaluación estarán dados por la aplicación de las funciones sobre todos las posibles instancias del grafo ponderando los resultados por la probabilidad de ocurrencia de cada uno de estas instancias del grafo. Este método, a pesar de retornar el

resultado exacto del problema, tiene como desventaja la imposibilidad de ser aplicado para grafos de tamaño mediano a grande, e incluso para grafos pequeños el tiempo de cálculo es inaceptable.
Debido a estas contrariedades es que, además de implementar este método exacto (que denominaremos Generación Completa de Estados), implementaremos dos métodos probabilísticos más: el método de Monte Carlo y una variante de este, denominada Monte Carlo Antitético generalizado, que busca la una reducción de la varianza del resultado obtenido.

En los siguientes puntos describiremos el diseño de estos tres métodos que se implementaron, todos ellos tomando como base la evaluación de un conjunto de instancias particulares del grafo de entrada utilizando las funciones de evaluación definidas en la sección E.4.2.

E.4.3.1. Generación Completa de Estados (GCE)

Como mencionamos arriba, este método retorna el resultado exacto de la evaluación de las funciones objetivos sobre el grafo ingresado, a costa de la evaluación de cada uno de los posibles estados que puede tomar el grafo, los cuales crecen super-exponencialmente respecto al número de nodos del grafo.

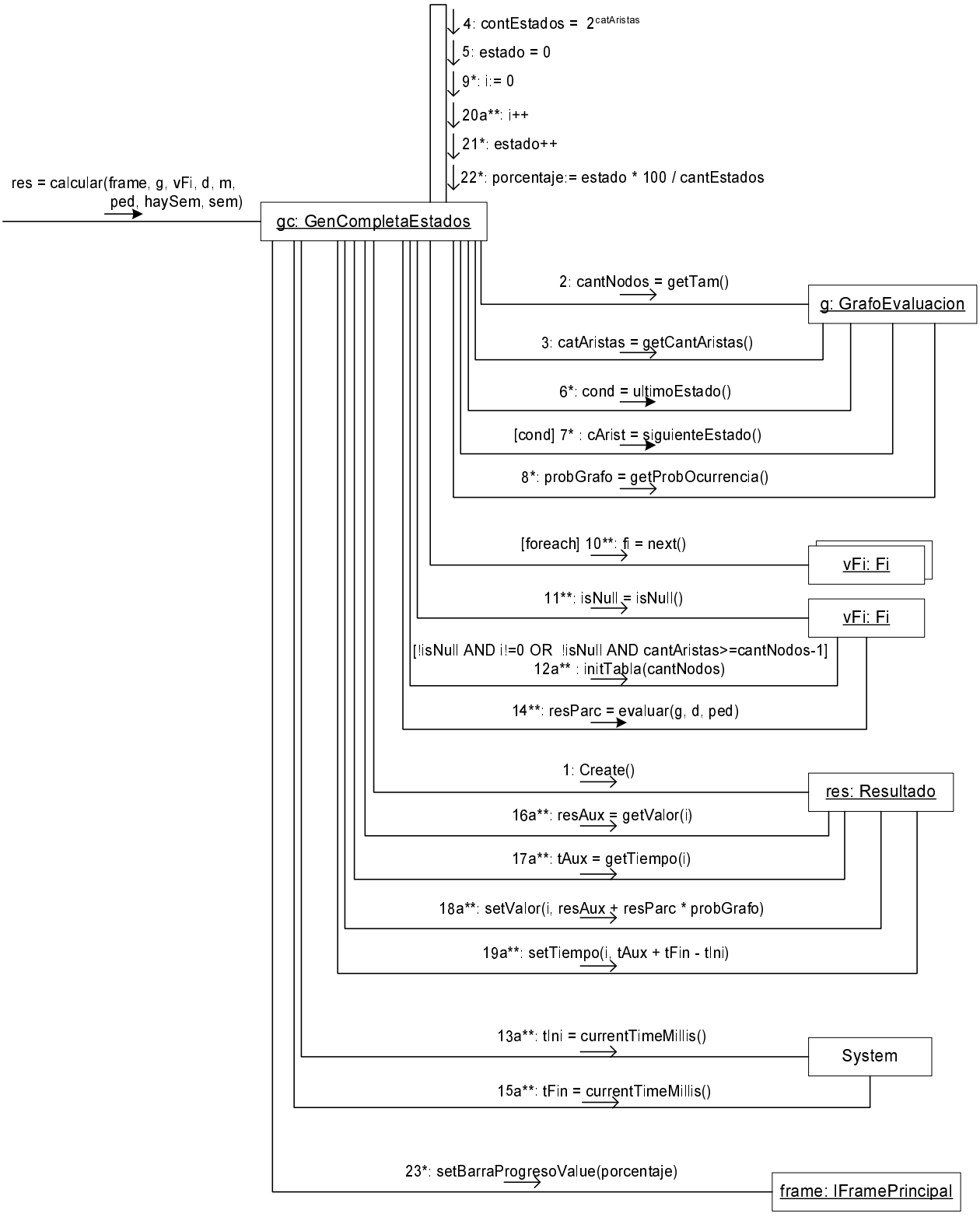


Fig. E.4: Diagrama de colaboración – GenCompletaEstados:calcular()

E.4.3.2. Monte Carlo Crudo (MCC)

El primero de los métodos probabilísticos implementado es el Monte Carlo Crudo o Monte Carlo vainilla.

La necesidad de utilizar técnicas de simulación como esta, es de imperiosa necesidad debido a que por ejemplo, el problema de evaluación de la confiabilidad diámetro acotada en el caso general es un problema NP-complejo, lo que casi elimina las posibilidades de encontrar algoritmos de complejidad polinominal [CP 03].

El método de MCC es la forma más básica de los métodos Monte Carlo: se sortean un conjunto de M instancias del grafo genérico, se evalúan las mismas y finalmente se hace una media de los resultados obtenidos ponderada según la probabilidad de ocurrencia de cada una de las instancias sorteadas.
Al aumentar el valor de M el método tiende a aproximarse al valor exacto, aumentando también el tiempo de procesamiento, por lo que es importante para que la utilización de esta técnica tenga los resultados esperados en la disminución de los tiempos de cálculo sin una pérdida muy grande de la calidad del resultado obtenido, la correcta elección del valor de M.
Dado que el resultado obtenido siempre (a menos de algún caso con propiedades particulares) difiere del valor exacto de la solución, es muy importante determinar alguna medida de la calidad de resultado. En nuestro caso hemos optado por la opción natural de calcular la desviación estándar de la solución obtenida, de forma de poder obtener un intervalo de confianza para el valor calculado.

En lo que refiere al diseño en sí del método, el mismo no agrega mayores novedades: se van generando instancias del grafo genérico (implementado por la clase *GrafoEvaluacion*), se evalúan las funciones objetivo (almacenadas en un arreglo de *Fi*) y se van almacenando los resultados y los sigmas obtenidos en una instancia de la clase *Resultado*. Este procedimiento se repite tantas veces como el valor de M que se indique como parámetro de entrada
La instancia de *Resultado* que se retorna, contendrá el resultado de la evaluación de las funciones objetivos indicadas en el parámetro de entrada 'vFi' junto con sus correspondientes sigmas, y será pasada a la clase que invocó el método que seguramente sea parte de la capa de presentación de la aplicación y se encargará del despliegue de estos resultados.

A continuación adjuntamos el diseño que hemos hecho para la implementación del método de Monte Carlo Crudo, para la evaluación de las funciones objetivo definidas:

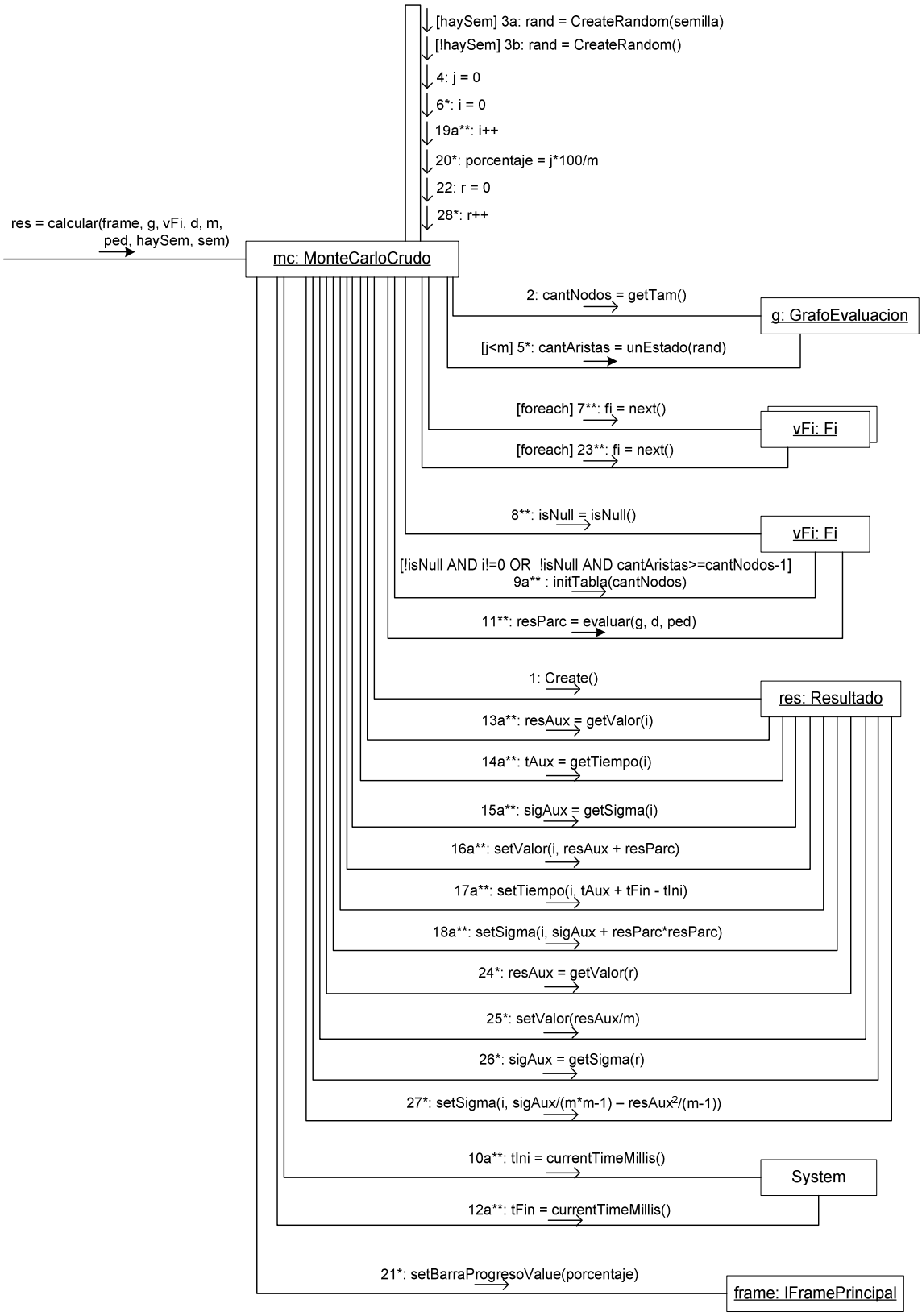


Fig. E.5: Diagrama de colaboración – MonteCarloCrudo:calcular()

E.4.3.3. Monte Carlo Antitético (MCA)

El MCC, a pesar de cumplir de buena forma el objetivo de obtener una buena aproximación de lo que sería el resultado exacto de la evaluación de las funciones objetivo sobre el grafo

genérico, adolece de la carencia de que la desviación estándar de este resultado tiende a ser un valor no muy pequeño para valores de M no muy grandes.
En pos de buscar un “mejor” resultado para las evaluaciones que se realicen, se encontraron varias técnicas de reducción de varianza para el método de Monte Carlo, entre ellas las planteadas en [CE 98] y en [CU 00], decidiendo finalmente utilizar el método de Monte Carlo Antitético generalizado [ER 92].
El diseño seguido se basa completamente en el diseño sugerido por Rubino y El Khadiri en el paper antes mencionado, por lo que sugerimos referirse a ese documento por detalles del mismo.

Cabe mencionar que la utilización de técnicas de reducción de varianza como la que implementa el MCA por lo general no son gratuitas y en la mayoría de los casos el costo que agregan se da en un aumento de los tiempos de cálculo. Esta hipótesis planteada será verificada luego experimentalmente, y será también una variable a tener en cuanta para la posterior elección del método de evaluación a utilizar en la etapa de optimización del proyecto.

E.4.4. Diagrama de Clases de Diseño

Para cerrar lo que es el capítulo de diseño de la aplicación, presentaremos el Diagrama de clases de diseño de la misma.

A continuación presentamos el Diagrama de Clases de lo que sería la capa de presentación de la aplicación. Como se notará, la misma se comunica con la capa lógica a través de la clase *ThreadCalculo* la cual implementa la interfaz *Runneable* para poder correr como un hilo y permitir la actualización de la barra de progreso y la implementación de un botón para cancelar la ejecución.

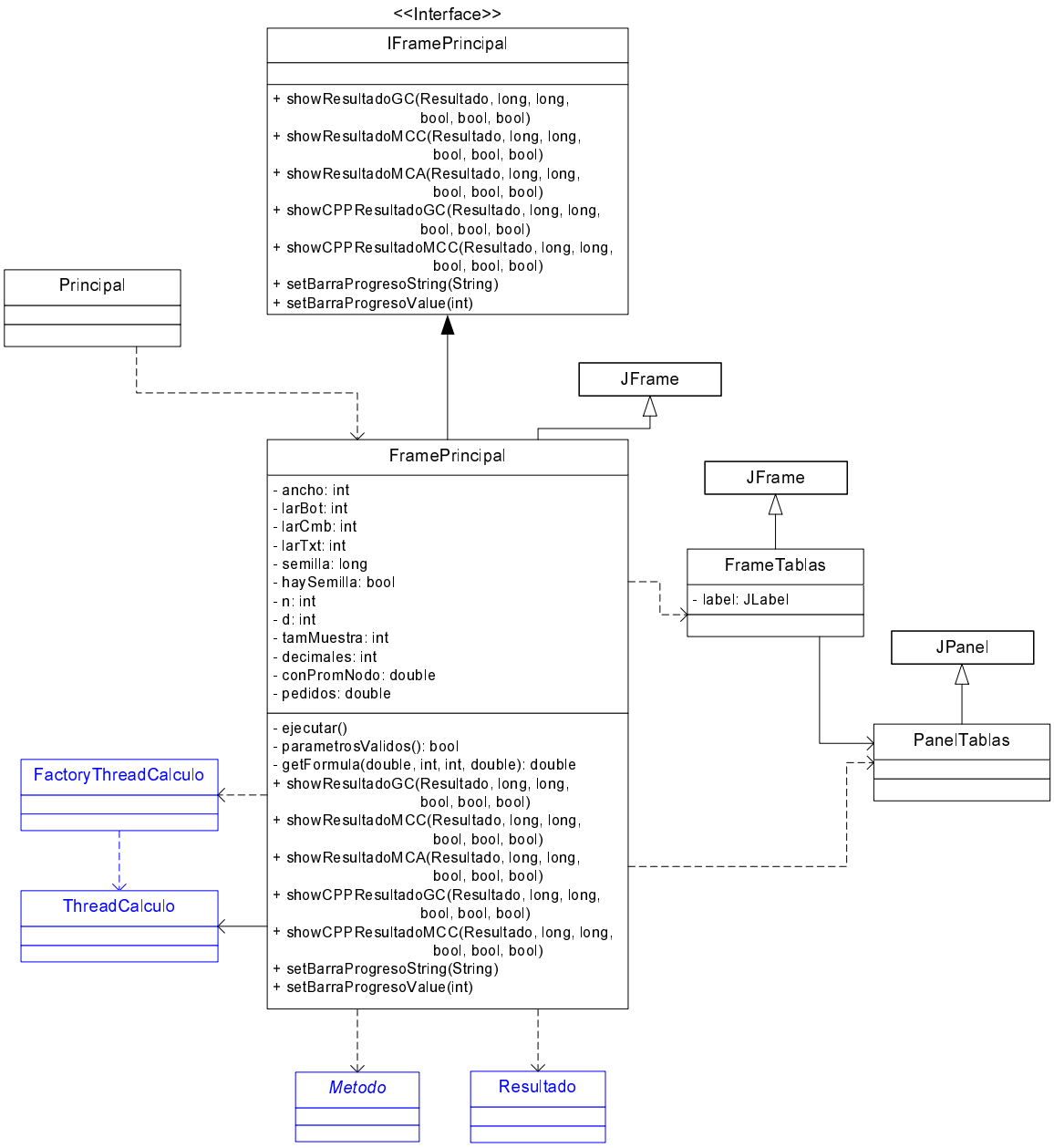


Fig. E.6: Diagrama de clases de diseño del evaluador (Capa de presentación)

Este último diagrama que presentamos a continuación muestra la capa lógica de la aplicación, en la que podemos observar lo que mencionábamos más arriba: la existencia de una interfaz para la comunicación de eventos producidos en la capa lógica a la capa presentación.

Básicamente (como lo muestran las operaciones de la interfaz *IFramePrincipal*) estos eventos corresponden al despliegue de los resultados obtenidos por cada uno de los métodos y la actualización de la barra de progreso.
Esta capa podría dividirse internamente en tres sub-capas:

- *ThreadCalculo* funcionaría como interfaz de acceso a las operaciones de la capa, que básicamente consisten en la ejecución de los métodos implementados.
- A su vez *ThreadCalculo* utiliza los servicios que le brinda la clase *Metodo*, superclase de todos los métodos implementados, los cuales tienen que implementar la operación *calcularGen()* la cual se encarga de la implementación del método correspondiente.
- Finalmente *calcularGen()* utilizará en todos los casos las implementaciones de las funciones objetivos que especifiquen a la clase abstracta *Fi*. En este caso las subclases deberán implementar la operación *evaluar()* que recibirá como parámetro una instancia particular del grafo genérico ingresado y realizará la evaluación de la función que corresponda sobre esa instancia.

También cabe mencionar la utilización de diferentes Data Types (clases auxiliares cuyas instancias no tienen identidad) en los diferentes puntos del procesamiento. Entre estas podemos mencionar:

- *GrafoEvaluacion*: Representa la información tanto de un grafo genérico como de una instancia particular del mismo. Este acoplamiento de la clase del grafo con las instancias que se generan a partir de la misma se realiza para poder generar secuencialmente todos los estados del grafo, lo cual nos es de particular utilidad para el método GCE.
- *Resultado*: Se encarga de almacenar la información referente a los resultados de las diferentes funciones objetivos y sus correspondientes sigmas en los casos que tenga sentido (métodos probabilísticos) junto con los tiempos de ejecución.
- *Tabla*: TAD que representa una matriz cuadrada de tamaño variable en la que cada uno de sus elementos posee las propiedades que se definan en el Data Type *Elemento.* Se utiliza de manera auxiliar para el almacenamiento de información de la instancia del grafo durante la implementación de los algoritmos probabilísticos.
- *Cola* y *ColaCarga*: TAD que implementan colas circulares y difieren en la información que se almacena en cada uno de los elementos de las mismas. Se usan el las implementaciones particulares de las *Fi*'s y específicamente en los recorridos BFS que se realizan de los nodos del grafo particular que se procese.

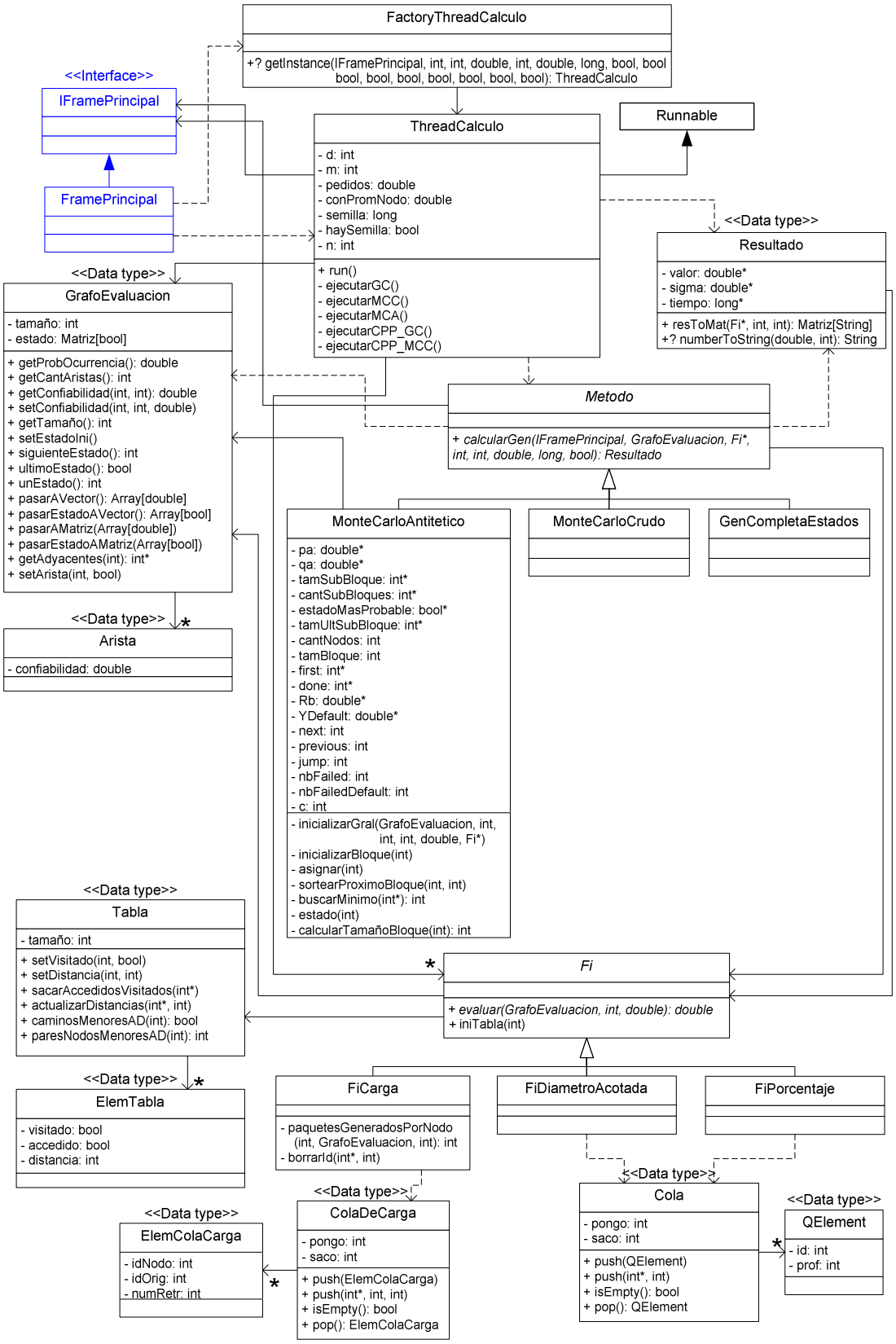


Fig. E.7: Diagrama de clases del evaluador (Capa lógica)

E.5. Validación y análisis de resultados

E.5.1. Introducción

Luego de haber presentado el modelo sobre el que se trabajaría, de haber hecho el análisis de lo métodos a implementar, y de haber explicado en detalle todo el desarrollo de la aplicación encargada de la evaluación de las tres funciones de interés sobre grafos, para culminar este apéndice referente a toda la etapa de evaluación en nuestro proyecto solo nos resta la validación de la herramienta implementada y el análisis de los resultados obtenidos, puntos éstos sobre los que nos extenderemos en la presente parte del apéndice.

La base para la realización de las pruebas es la realización de corridas de la aplicación para conjuntos de parámetros específicos, elegidos de forma particular según el tipo de prueba que se efectúe. Denominaremos caso de prueba a cada uno de estos conjuntos de parámetros, y lo identificaremos de forma unívoca con un nro. de caso, de forma de poder referirnos fácilmente a cada uno de ellos.
En todas las pruebas realizadas es importante tener en mente que los resultados obtenidos son relativos. O sea, que tomamos un conjunto representativo de casos de pruebas para la realización de la validación o el análisis en cuestión, y a partir de los resultados obtenidos en los mismos sacamos conclusiones que tomamos como generales sin perder de vista que formalmente no los son y solamente son válidas para el conjunto de casos utilizados. Esta limitante es implícita al tipo de pruebas que efectuaremos, las cuales son de características empíricas y no formales.

Dividiremos las pruebas en dos grandes conjuntos: aquellas utilizadas para la comprobación empírica de algunas propiedades de correctitud y exactitud de resultados de las implementaciones realizadas, y aquellas concernientes al estudio de los resultados que se obtienen a la búsqueda de propiedades interesantes en los mismos.
Las primeras buscan la comprobación de que los algoritmos y métodos implementados son correctos a través de la comparación con resultados conocidos, ya sea de proyectos anteriores u obtenidos teóricamente, y de la comparación entre los diferentes algoritmos implementados en busca de un grado de consistencia importante entre los mismos.
Las segundas pruebas, referentes al análisis de los resultados, tienen el objetivo de buscar comportamientos interesantes en los resultados que se obtienen que permitan sacar conclusiones novedosas de los métodos implementados.
Para un correcto desarrollo de todas estas pruebas, las mismas fueron ordenadas de forma que en cada una de ellas puedan ser utilizados, en caso de ser necesario, las conclusiones obtenidas en las anteriores.
En definitiva, las pruebas que realizaremos abarcan los siguientes puntos y siguen el orden en que se presentan:

- Verificación del funcionamiento del método GCE.
- Verificación de los métodos Monte Carlo implementados.
    - La solución exacta pertenece a un intervalo de confianza obtenido producto de la evaluación mediante Monte Carlo.
    - El tamaño del intervalo disminuye en torno al valor exacto a medida que se aumenta el número de muestras.
    - Evolución de las soluciones obtenidas al variar cada uno de los parámetros de entrada manteniendo fijos los restantes.
- Estudio de casos de borde y casos no válidos.
- Comparación de los métodos Monte Carlo entre sí.
- Análisis de los resultados obtenidos en las implementaciones en C++ y Java.

Es de interés aclarar que todas las pruebas realizadas son reproducibles, ya que para aquellas que involucran la generación y utilización de variables aleatorias, las mismas son obtenidas mediante una semilla que es un parámetro de entrada de la aplicación y coincide con el número de caso con que hemos identificado cada una las pruebas.

Caben algunas aclaraciones respecto a las pruebas que presentaremos y a como el lector puede acceder a toda la información referente a las mismas. Las pruebas individuales realizadas fueron identificadas por un número de caso que sirve de clave al conjunto de parámetros de entrada que definen la prueba. La aplicación de evaluación no recibe como entrada un grafo en sí, sino que construye el mismo a partir de los parámetros que precisamente agrupamos bajo el número de caso. Estos parámetros son la cantidad de nodos del grafo (N), la cantidad de conexiones permitidas en promedio a los nodos del grafo (CCP), el tiempo de vida máximo para los paquetes que circulan en la red (D), y la cantidad de pedidos que llegan a cada nodo en promedio (CPN).
Además, en los casos en que aplica, se utiliza este número de caso como valor para la semilla del generador de números aleatorios de la aplicación, haciendo reproducible cualquiera de los casos que se presentarán.

En las siguientes secciones se presentan cuadros, gráficas y comentarios correspondientes a cada una de estas pruebas.

E.5.2. Verificación del método GCE

La primera de las pruebas que realizaremos será la concerniente a la verificación del correcto funcionamiento de los métodos implementados mediante la generación completa de estados (GCE).
Esto tiene básicamente una justificación: en caso de comprobarse la correctitud de este método, luego los resultados obtenidos mediante el mismo podrán ser utilizados como valores de comparación para la comprobación de la correctitud de los métodos Monte Carlo, por tratarse éstos de valores exactos.

Las pruebas que se realizarán consisten en la comparación de los resultados producto de la ejecución de la aplicación, con resultados conocidos de proyectos anteriores. Estos resultados son únicamente de confiabilidad clásica, por los que las comparaciones son todas para el caso particular $D = N\text{-}1$.
Debido a este mismo hecho solo podemos hacer comparaciones de los resultados de la confiabilidad, no pudiéndose hacer lo mismo con la carga y el porcentaje de nodos conectados. Esta limitación en último término, no es de gran importancia debido a que la base del proyecto es el estudio y la optimización de la confiabilidad. De todos modos podría ser tomada como una validación del algoritmo de carga, la comparación de los resultados para el mismo con la formula usada para el algoritmo de optimización (ver **Apéndice E**).

A continuación presentamos el cuadro con los resultados obtenidos para grafos completos:

Caso	Entrada						GCE	
	N	CCP	D	CPN	M	Sol.	GCE	Dif.
26	5	1,8	4	1	512	0,6058	0,6058	0
27	6	2,25	5	1	16384	0,7171	0,7171	0
28	7	2,7	6	1	20971	0,8108	0,8108	-1,80E-11

Tabla E.1: Resultados para validación de la GCE – Grafos completos

Las columnas presentadas corresponden, en primer término a los parámetros de entrada con que se ejecutó el programa; luego (columna '*Sol.*') a la solución conocida de proyectos anteriores; y finalmente al valor obtenido producto de la ejecución ('*GCE*') y la diferencia entre éste y la solución conocida ('*Dif.*')
Como se notará, para los casos utilizados, las diferencias son nulas o despreciables, lo que permite pensar en el buen comportamiento del método GCE, al menos para los casos con las particularidades señaladas.

De la misma forma, se hicieron algunas pruebas con grafos no completos (red Montevideo y UDL) cuyos resultados se resumen en la siguiente tabla:

Caso	Entrada						GCE	
	N	r	|E|	E	M	Sol.	GCE	T
57	14	0,8	21	Mon	100000	0,6389	0,6389	60
58	10	0,9	20	UDL	100000	0,9990	0,9990	119

Tabla E.2: Resultados para validación de GCE – Grafos no completos

Para estas dos redes con que se probó el programa, se tienen resultados de confiabilidad conocidos de un trabajo previo [CRU 01]. Para el caso de la red UDL, en el paper original se menciona como valor de confiabilidad el valor 0,959898 lo que difiere con el valor calculado con nuestros algoritmos. Esta diferencia es producto de un error en la publicación del mismo, hecho que se verificó con uno de los autores del mismo.

En la tabla E.2 se presentan los dos casos de redes no completas estudiadas, junto con la solución que presenta el paper original para la misma (columna '*Sol.*'), la solución obtenida mediante nuestra aplicación (columna '*GCE*'), y el tiempo de ejecución medido en segundos para esta última (columna '*T*')

E.5.3. Verificación de métodos Monte Carlo

En esta sección desarrollaremos una serie de pruebas para, en una primera etapa, la validación de los métodos no exactos implementados (Monte Carlo Crudo -MCC- y Monte Carlo Antitético generalizado -MCA-), y el posterior análisis de sus resultados.

Para ello comenzaremos con la validación de la correctitud de los métodos, la cual realizaremos en dos etapas: contra resultados conocidos, y contra resultados obtenidos a través del método GCE. Continuando en lo que es la validación, comprobaremos el aumento en la calidad de las soluciones al aumentar el número de muestras usadas en la simulación. Finalmente haremos algunas pruebas tendientes a comprobar el comportamiento de las soluciones en función del comportamiento de cada uno de los parámetros de entrada.

Pasando a lo que es el análisis de los resultados, el mismo se concentrará en las mismas pruebas utilizadas para la validación del comportamiento de las soluciones al variar individualmente cada uno de los parámetros de entrada.

En las siguientes sub-secciones iremos detallando cada una de estas pruebas que acabamos de presentar.

E.5.3.1 Validación de los métodos

Comenzaremos entonces con la etapa de validación de los métodos, haciendo la prueba básica sobre los mismos: la comprobación de que los resultados obtenidos son correctos.
Dado que estamos trabajando con métodos estadísticos, los resultados obtenidos no son exactos y vienen cargados de un error determinado. A partir de la estimación de este error en los resultados, podemos definir un intervalo con centro en el valor obtenido y en donde es conocida la probabilidad de que se encuentre el resultado exacto.
En base a esto las pruebas de correctitud de los resultados del MCC y el MCA están centradas en la comprobación de que el resultado que tomamos como exacto (que puede ser conocido de proyectos o publicaciones anteriores, o producto de la ejecución del GCE para el mismo conjunto de parámetros) está dentro del intervalo [sol – 3sigma, sol + 3sigma], en donde sigma mide la desviación estándar del resultado

Dividiremos las pruebas precisamente en estos dos grupos que mencionábamos, comenzando con la comparación con resultados conocidos y siguiendo con la comparación con resultados del GCE.

E.5.3.1.1. Comparación con resultados conocidos

Dado lo novel del concepto de confiabilidad diámetro acotada, lo resultados que se pueden obtener de proyectos y trabajos existente son escasos o nulos. Debido a esto, al igual que en el caso de la validación de la GCE, restringiremos la validación mediante estas comparaciones al caso particular de grafos en los que $D = N-1$, para los que la confiabilidad diámetro acotada se reduce al concepto de confiabilidad clásica.
Estas pruebas, por ende, estarán reducidas a la validación de la función de confiabilidad diámetro acotada, quedando para validarse las demás funciones mediante comparación con resultados obtenidos mediante GCE.

Caso	Entrada						MCC		MCA	
	N	CCP	D	CPN	M	Sol.	MCC	Sig	MCA	Sig
29	8	3,15	7	1	26843	0,8793	0,8789	0,0020	0,8791	0,0019
30	9	3,6	8	1	687194	0,9251	0,9254	0,0003	0,9248	0,0003
31	10	4,05	9	1	351843	0,9542	0,9542	0,0004	0,9540	0,0003

Tabla E.3: Validación de MCC y MCA comparando con resultados conocidos en redes completas.

Caso	Entrada						MCC				MCA				
	N	r	|E|	E	M	Sol.	MCC	Sig	T	Cont sol.	MCA	Sig	T	Cont sol.	Sem
57	14	0,8	21	Mon	100000	0,6389	0,6381	0,0015	26	Sí	0,6388	0,0012	55	Sí	1
58	10	0,9	20	UDL	100000	0,9990	0,9991	0,0001	30	Sí	0,9989	0,0001	32	Sí	1
59	21	0,99	26	Arpanet	100000	0,9974	0,99723	0,00016	102	Sí	0,997559	0,00015	55	Sí	1

Tabla E.4: Validación de MCC y MCA comparando con resultados conocidos en redes no completas.

La presentación de los resultados sigue el mismo formato presentado en la Tabla E.1: las primeras columnas muestras los parámetros utilizados, luego la sigue el valor conocido de confiabilidad para el grafo, y finalmente los valores obtenidos mediante el MCC y el MCA junto a sus respectivas desviaciones estándar (columna '*Sig*')

En definitiva, en todos los grafos analizados, el valor exacto está dentro del intervalo de confianza definido, comprobándose la correctitud de los métodos para estos casos, y dándonos un margen para la extensión de esta conclusión de correctitud para todos los casos.

E.5.3.1.2. Comparación con resultados obtenidos en GCE

Siguiendo con las pruebas de validación de los métodos estadísticos, pasaremos a hacer comparaciones de lo resultados obtenidos mediante éstos con los resultados de la GCE.
Esta prueba plantea algunas limitaciones como validación de las implementaciones realizadas para el MCC y el MCA, dado que el algoritmo GCE solo puede ser utilizado para grafos de hasta 7 nodos debido a los tiempos que generarían corridas para grafos más grandes.

En este caso, además de presentar el conjunto de parámetros con que se ejecutó la aplicación, y los resultados obtenidos para cada uno de los tres métodos, se presentan de forma explícita los intervalos de confianza para los métodos estadísticos, contra los cuales comprobaremos la inclusión del resultado obtenido mediante la GCE.

Las pruebas ahora sí pueden hacerse para las tres medidas de interés sobre los grafos, presentándose consecutivamente los resultados obtenidos para la confiabilidad diámetro acotada (Tabla E.5), el porcentaje de nodos conectados por caminos de largo menor o igual a *D* (Tabla E.6), y la carga en la red (Tabla E.7).

	Entrada					Diámetro Acotada							
Caso	N	CCP	D	CPN	M	GCE		MCC			MCA		
						Val	T(seg.)	Limite inf.	Limite sup.	T(seg.)	Limite inf.	Limite sup.	T(seg.)
1	3	1	2	1	8	0,5	0,00	0,0763	1,1737	0,00	0,5	0,5	0,016
2	3	1	1	1	8	0,125	0,00	0	0	0,00	0	0,5	0,00
3	4	2	2	1,5	32	0,724	0,02	0,4374	0,9366	0,02	0,4521	0,9219	0,00
4	4	2	3	1,5	32	0,855	0,00	0,5169	0,9831	0,02	0,7334	1,0166	0,00
5	4	1	3	1,5	32	0,275	0,00	0	0,4406	0,00	0,1436	0,3884	0,00
6	5	3	1	0.5	512	0,05631	0,03	0,021644	0,079916	0,06	0,026579	0,078881	0,00
7	5	2	3	0.5	512	0,6523	0,08	0,55871	0,68729	0,03	0,60406	0,69274	0,03
8	5	1	3	0.5	512	0,1159	0,02	0,05667	0,13473	0,00	0,08148	0,14892	0,00
9	6	4	4	2	16384	0,998	1,91	0,9969653	0,9990347	1,72	0,9961012	0,9984988	1,58
10	6	3	2	2	16384	0,6034	1,33	0,590025	0,612975	1,07	0,592816	0,610984	1,19
11	6	2	2	4	16384	0,13	1,59	0,117639	0,133161	0,34	0,125786	0,140414	0,95
12	7	2	3	3	20971	0,2792	188,60	0,275155	0,293845	0,63	0,275452	0,289348	1,00
13	7	3	2	1	20971	0,3225	112,20	0,311825	0,331175	1,35	0,312975	0,327825	1,35
14	7	4	2	1	20971	0,8078	109,80	0,801164	0,817436	2,35	0,798688	0,813712	2,22
15	7	5	1	1	20971	0,02173	23,57	0,019028	0,025112	0,783	0,0187854	0,0247146	0,63

Tabla E.5: Validación de CDA en MCC y MCA comparando con resultados de GCE.

Como se esperaba, en todos los casos planteados los intervalos definidos por un entorno de 3 sigma centrado en la solución retornada por el MCC y el MCA contienen el valor obtenido mediante GCE, lo que comprueba la corretitud de los algoritmos implementados al menos para este conjunto de casos de prueba.

	Entrada					Porcentaje							
Caso	N	CCP	D	CPN	M	GCE		MCC			MCA		
						Val	T(seg.)	Limite inf.	Limite sup.	T(seg.)	Limite inf.	Limite sup.	T(seg.)
1	3	1	2	1	8	62,5	0,00	26,79	114,87	0,00	49,992	74,988	0,00
2	3	1	1	1	8	49,9	0,02	19,52	72,08	0,00	NAN	NAN	0,00
3	4	2	2	1,5	32	89,7	0,02	77,43	99,57	0,02	86,32	96,88	0,00
4	4	2	3	1,5	32	91,9	0,00	74,66	99,14	0,00	83,13	101,07	0,00
5	4	1	3	1,5	32	49,5	0,02	25,25	62,15	0,00	43,16	54,44	0,00
6	5	3	1	0.5	512	75	0,11	73,7822	77,4578	0,00	75	75	0,09
7	5	2	3	0.5	512	84,76	0,03	79,965	86,595	0,05	81,6425	86,9375	0,05
8	5	1	3	0.5	512	41,72	0,08	40,146	47,934	0,08	40,6266	43,3134	0,03
9	6	4	4	2	16384	99,93	1,08	99,89571	99,96429	1,44	99,87097	99,94903	1,33
10	6	3	2	2	16384	93,28	1,94	93,10285	93,63715	1,50	93,03155	93,48845	1,40
11	6	2	2	4	16384	70,12	2,79	69,6078	70,6722	1,41	69,5867	70,3133	4,11
12	7	2	3	3	20971	74,47	212,70	73,8825	74,9175	1,568	74,1476	74,7524	1,57
13	7	3	2	1	20971	88,13	200,40	87,885	88,455	2,12	88,05045	88,40955	2,03
14	7	4	2	1	20971	98,23	199,40	98,09794	98,30206	2,55	98,12379	98,31621	2,63
15	7	5	1	1	20971	83,33	148,8	83,18993	83,53007	2,575	NAN	NAN	2,77

Tabla E.6: Validación del porcentaje de nodos conectados en MCC y MCA comparando con resultados de GCE.

En este caso nuevamente es notorio que los intervalos definidos incluyen a las soluciones obtenidas mediante GCE.
Cabe aclarar el por qué de los valores “NaN” obtenidos resultado de la ejecución del MCA para los casos de prueba número 2 y 15. Analizando en detalle la ejecución para los conjuntos de parámetros definidos en los casos mencionados, se notó que los mismos se producen a problemas de precisión en las variables utilizadas que derivan en la realización de una raíz cuadrada de un número negativo en un momento del cálculo del porcentaje de nodos conectados.

Finalmente para la validación de la carga se utilizó el mismo conjunto de pruebas que en los casos anteriores, pero se agregó una nueva columna con el resultado obtenido para la carga a través de la función derivada para la estimación de la misma (ver sección E.2.2.2, más arriba en este mismo documento)

	Entrada					Carga								
Caso	N	CCP	D	CPN	M	GCE		MCC			MCA			Formula
						Val	T(seg.)	Limite inf.	Limite sup.	T(seg.)	Limite inf.	Limite sup.	T (seg.)	
1	3	1	2	1	8	4,5	0,00	1,6061	6,8939	0	3	6	0	4,5
2	3	1	1	1	8	3	0,00	1,172	4,328	0,00	3	3	0,00	3
3	4	2	2	1,5	32	28	0,00	21,02	38,18	0,00	21,31	32,89	0,02	27,9
4	4	2	3	1,5	32	49,3	0,02	21,62	51,98	0,00	39,67	77,53	0,00	49,2
5	4	1	3	1,5	32	12,6	0,00	4,38	22,62	0,00	8,89	16,51	0,00	12,7
6	5	3	1	0.5	512	7,5	0,03	7,37822	7,74578	0,00	7,5	7,5	0,31	7,5
7	5	2	3	0.5	512	23,75	0,02	19,7829	24,7371	0,05	22,8348	25,5852	0,03	23,75
8	5	1	3	0.5	512	5,781	0,03	5,0575	6,6985	0,03	5,2872	6,0468	0,03	5,781
9	6	4	4	2	16384	2290	5,26	2258,641	2309,359	9,07	2281,791	2296,209	9,08	2265
10	6	3	2	2	16384	122,3	1,30	121,553	123,647	0,55	121,5718	122,8282	0,48	122,4
11	6	2	2	4	16384	124,8	1,69	123,2787	126,3213	0,45	123,931	125,869	0,44	124,8
12	7	2	3	3	20971	228,6	241,20	225,895	233,305	1,64	227,5264	231,0736	1,27	228,6
13	7	3	2	1	20971	73,5	84,96	72,8925	74,0475	0,75	73,34928	73,65072	73,50	73,5
14	7	4	2	1	20971	121,3	82,66	120,8082	122,1918	1,14	120,4975	121,5025	1,05	121,3
15	7	5	1	1	20971	35	24,84	34,93857	35,08143	0,31	35	35	0,42	35

Tabla E.7: Validación de la carga en MCC y MCA comparando con resultados de GCE.

Nuevamente para todos los casos estudiados los resultados fueron los esperados, comprobándose el buen funcionamiento de la función de evaluación de la carga ejecutada mediante los métodos estadísticos, tanto en lo referente a la pertenencia del valor generado mediante GCE en el intervalo definido, como a la pertenencia del resultado obtenido a través de la fórmula en ese mismo intervalo.
Esto último también es una prueba de validación importante para la mencionada función de carga, que a pesar de tener las mismas limitaciones que hemos venido mencionando para la validación de los métodos (no puede generalizarse a todos los casos) resulta de interés para una futura decisión de la utilización de la misma en el algoritmo de optimización.

E.5.3.2. Validación del comportamiento de la solución al variar el número de muestras

Una vez desarrollada la validación de los métodos implementados y comprobado su correcto funcionamiento, es posible extender esta validación analizando el comportamiento de la misma. En esta primera etapa, tomaremos como base la propiedad de los métodos Monte Carlo de que al aumentar el número de muestras que se realicen del experimento en cuestión, la varianza del resultado final tiende a disminuir, lo que se deduce de las fórmulas de la sección E.2.3.2.

En definitiva, el procedimiento seguido fue tomar cuatro casos arbitrarios (casos 6, 7, 11 y 13) y realizando una serie de 10 experimentos en los que en cada uno se fue aumentando el número de muestras de forma lineal, comprobar el enunciado mencionado arriba de la disminución de la varianza.
Los resultados que se obtuvieron se muestran y comentan a continuación.

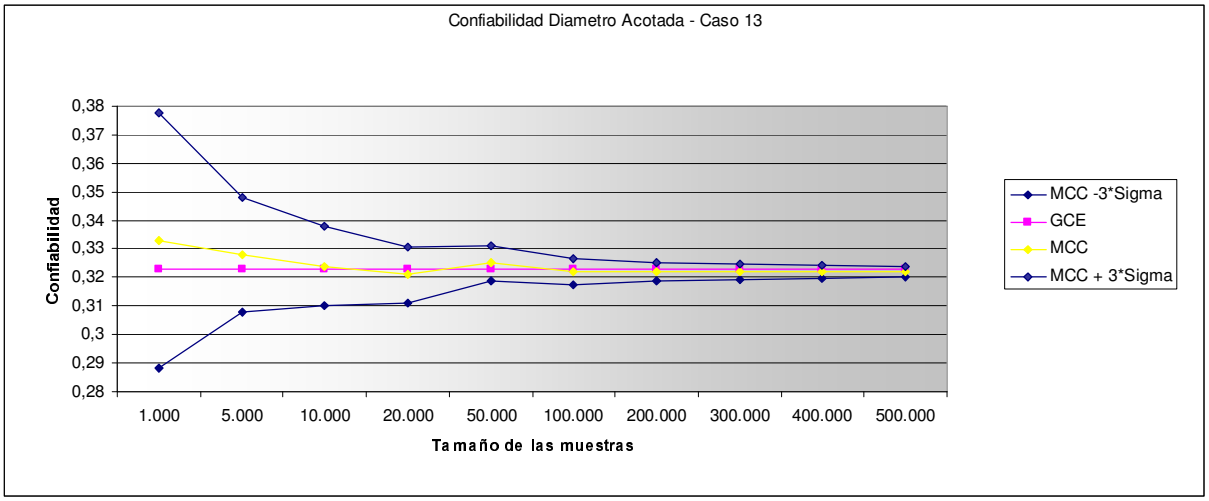
E.5.3.2.1. MCC

Como acabamos de mencionar, se realizó el experimento de correr varias veces el programa aumentando el número de muestras cada vez, con cuatro casos de prueba. Para no sobre cargar el documento, y dado que en todos los casos se observó el comportamiento esperado, decidimos incluir uno solo de estos casos que consideramos como representativo, en el presente documento.
El resto de los casos junto con las tablas de donde se obtuvieron los datos para las gráficas siguientes, serán incluidos en una planilla en el CD en el que se entrega como parte de la presentación del trabajo.
El caso presentado corresponde al siguiente juego de parámetros:

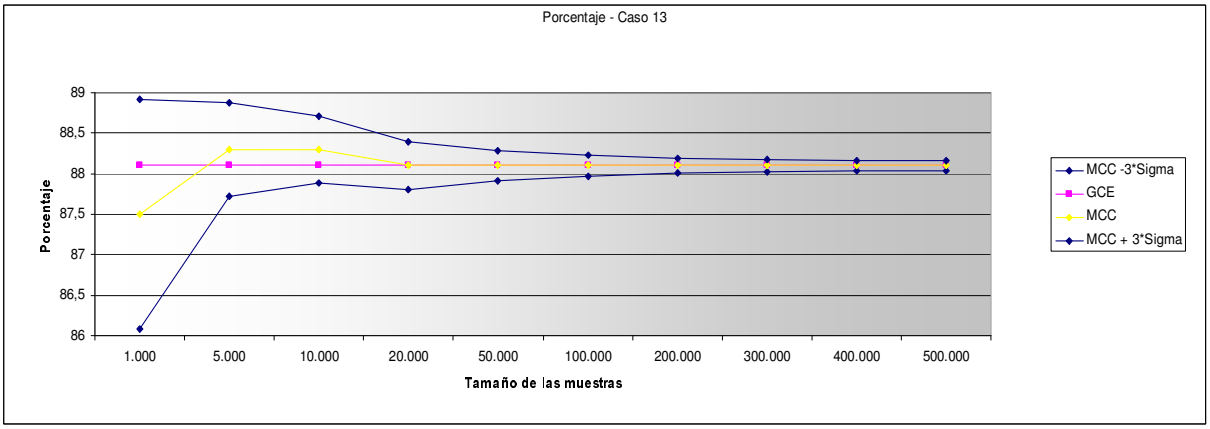
Caso	N	CCP	D	CPN
13	7	3	2	1

Tabla E.8: Parámetros para el caso de estudio del MCC

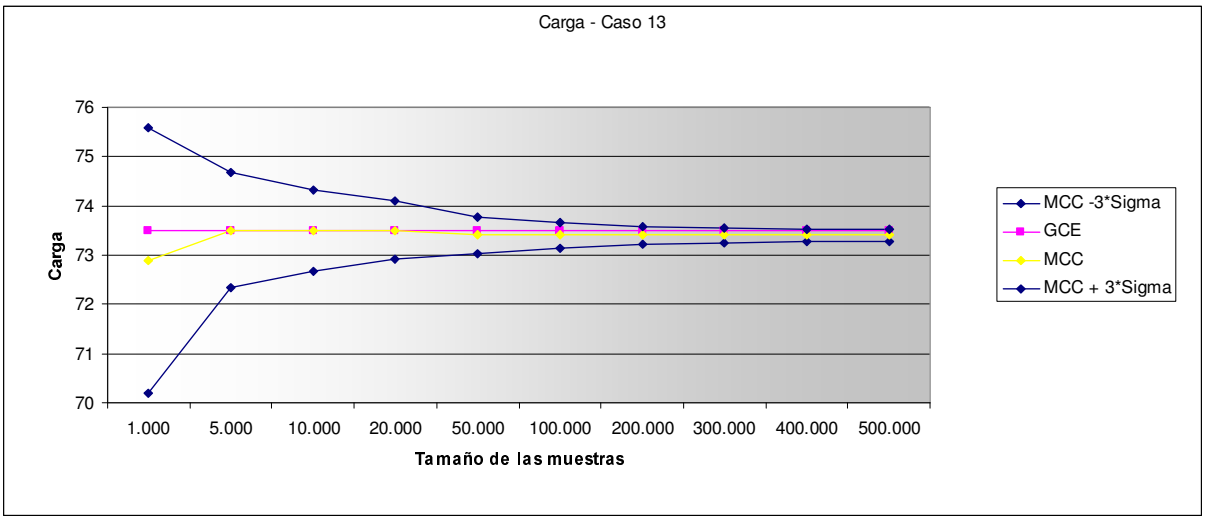
Las gráficas que se presentan corresponden a la variación del resultado obtenido mediante MCC para la confiabilidad diámetro acotada, el porcentaje de nodos conectados por caminos de largo menor que *D*, y la carga. A este resultado se le agrega el intervalo de amplitud de +- 3 sigma que definimos y el valor obtenido mediante GCE para cada una de estas funciones.
Los valores de M muestreados son: 1.000, 5.000, 10.000, 20.000, 50.000, 100.000, 200.000, 300.000, 400.000 y 500.000.



Grafica E.1: MCC - Variación de la varianza de la confiabilidad respecto al nro. de muestras



Grafica E.2: MCC - Variación de la varianza del porcentaje de nodos conectados respecto al nro. de muestras

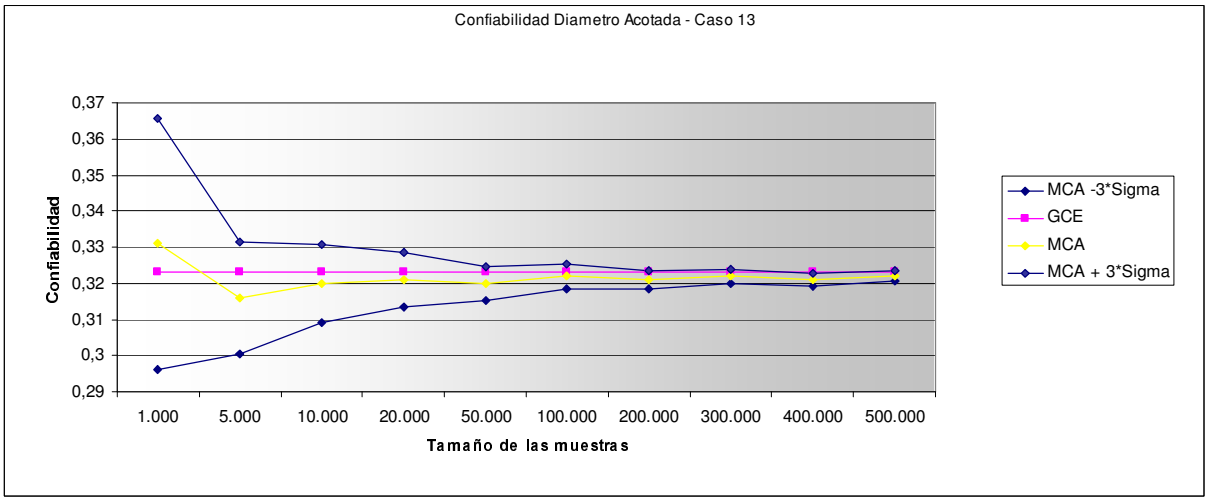


Grafica E.3: MCC - Variación de la varianza de la carga respecto al nro. de muestras

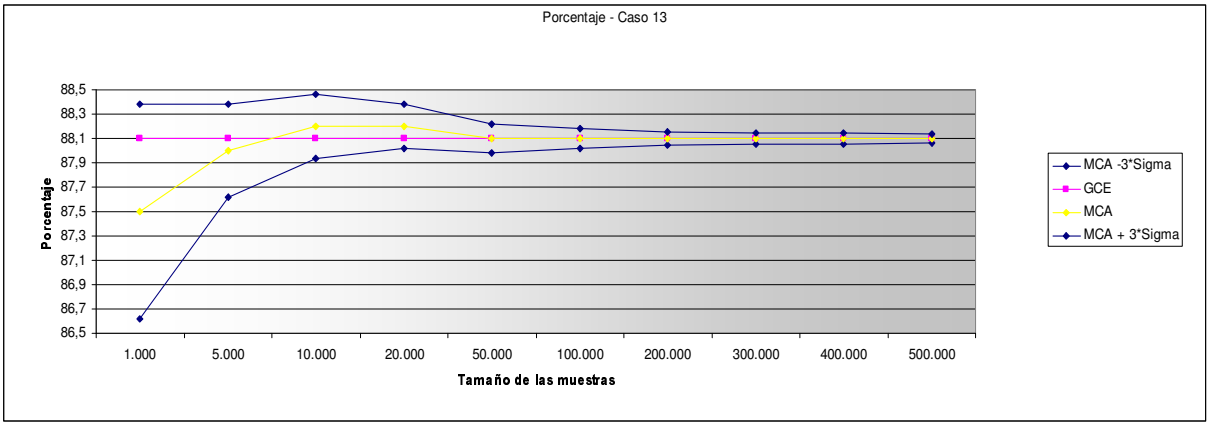
Como se nota en todas las gráficas, en todos los casos se aprecia una notoria disminución del intervalo de +- 3 sigma centrado en el valor resultado obtenido, como era de esperarse. Además se puede comprobar también que las soluciones continúan siendo de buena calidad, ya que el valor exacto (obtenido por el método GCE) continúa quedando incluido en el intervalo de confianza definido.

E.5.3.2.2. MCA

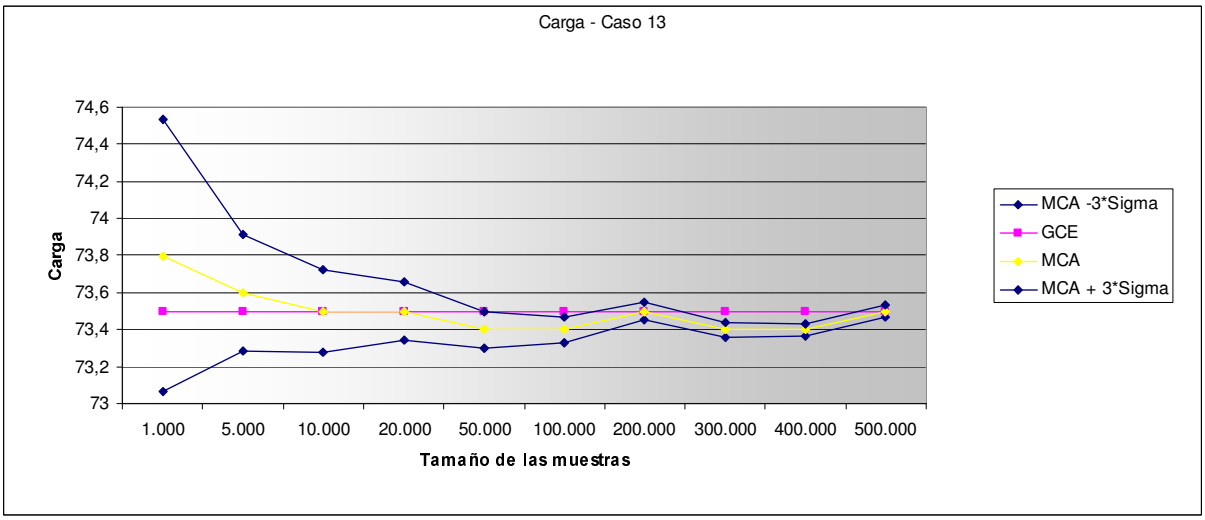
Para el método MCA, se repetirá el mismo experimento que para el MCC utilizando incluso el mismo caso de prueba. Nuevamente quedan disponibles en otro documento los resultados obtenidos para otros 3 casos y las tablas de datos para el presente.



Grafica E.4: MCA - Variación de la varianza de la confiabilidad respecto al nro. de muestras



Grafica E.5: MCA - Variación de la varianza del porcentaje de nodos conectados respecto al nro. de muestras



Grafica E.6: MCA - Variación de la varianza del porcentaje de nodos conectados respecto al nro. de muestras

En lo que respecta a la variación del valor del sigma ante el aumento en la cantidad de muestras tomadas, el comportamiento es el esperado en las tres funciones, notándose una disminución del mismo.

A diferencia del MCC, en el caso estudiado en el MCA se observa un comportamiento factible de darse, que corresponde con la no pertenencia del valor exacto para la función en el intervalo de confianza definido, el cual se nota claramente en la Gráfica E.6.

E.5.3.3. Validación y conclusiones del comportamiento de la solución al variar un parámetro

Visto y comprobado el correcto comportamiento de los métodos implementados para las tres funciones de interés sobre los grafos, podemos comenzar a hacer un análisis del comportamiento de los resultados que se obtienen ante ciertas condiciones en los parámetros de entrada y las variables de decisión del sistema.
Para esto el enfoque elegido es el siguiente: para cada uno de los parámetros de entrada realizamos una secuencia de corridas variando el parámetro en cuestión y manteniendo fijo el resto, observándose el comportamiento de las tres funciones en cada secuencia.

Para cada una de estas secuencias podemos prever un comportamiento determinado de cada una de las funciones según el modelo definido, lo cual pretendemos comprobar con el presente análisis. A continuación se presentan las gráficas obtenidas en cada caso, tanto para el MCC como para el MCA, junto un comentario comparando el comportamiento observado empíricamente con el que intuitivamente se esperaría según el modelo.

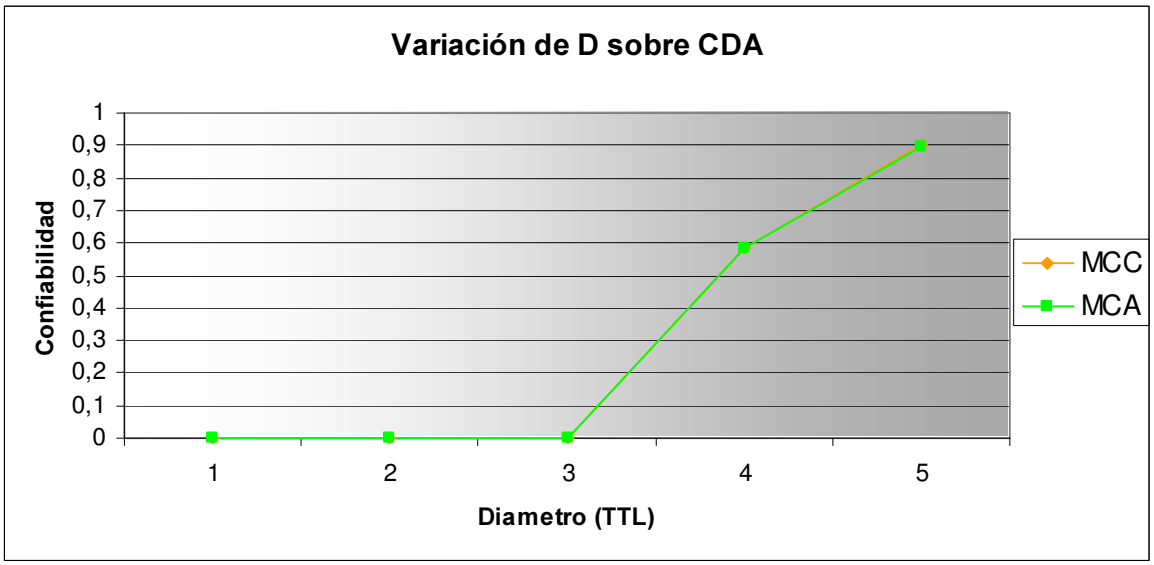
E.5.3.3.1. Variación de D.

Para empezar estudiaremos el comportamiento de las funciones al variar el largo máximo permitido para que un camino sea considerado como operacional en la red (D).
Se presentarán consecutivamente los resultados obtenidos para la confiabilidad diámetro acotada, el porcentaje y la carga.

Parámetros fijos: N = 50, CCP = 6, CPN = 1, M = 500

Caso	D	Diámetro Acotada					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)
32	1	0	0	0	0	0	0,047
33	2	0	0	0,112	0	0	0,93
34	3	0,002	0,002	0,203	0	0	0,25
35	4	0,584	0,022	3,11	0,584	0,0248	2,74
36	5	0,902	0,0133	4,36	0,895	0,013	4

Tabla E.9: Valores para CDA al variar D



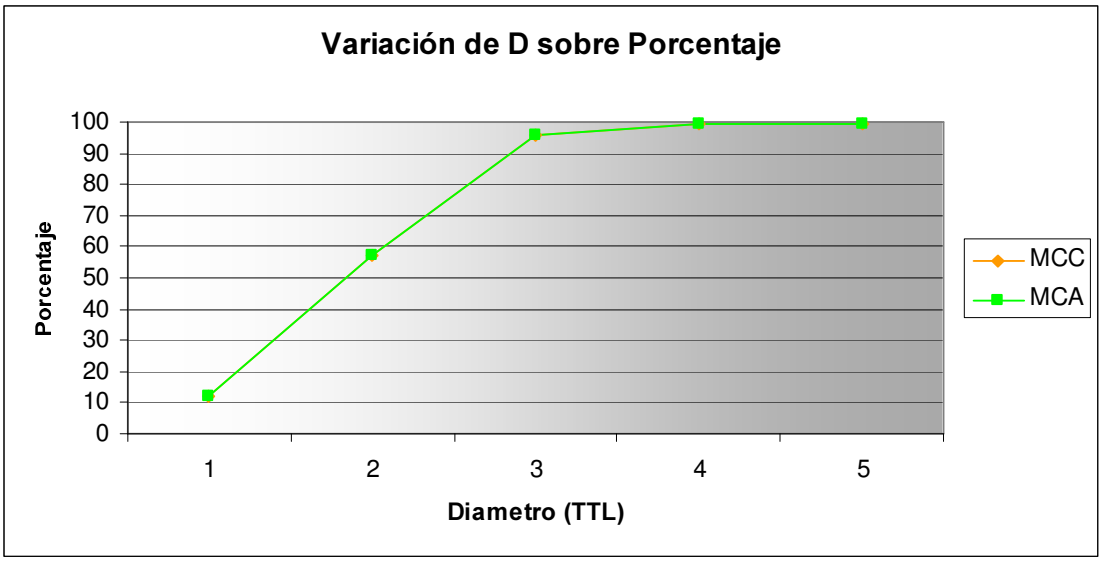
Grafica E.7: Comportamiento de la CDA al variar D

Intuitivamente es de esperar un aumento en la CDA al aumentar D debido a que caminos más largos aumentan la probabilidad de conectarse con más nodos en la red. Lo que aparece como novedoso es lo abrupto del cambio ya que un cambio de D=3 a D=4 generó un aumento de la CDA de 0 a más de 0,5. Este comportamiento que en la literatura ha sido observado en otros modelos de grafos aleatorios, podría ser interesante de estudiar en un análisis más detallado, que escapa al alcance de este proyecto.

Parámetros fijos: N = 50, CCP = 6, CPN = 1, M = 500

Caso	D	Porcentaje					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)
32	1	12,2	0,0417	1,2	12,2	0,00629	1,03
33	2	57,5	0,233	3,42	57,5	0,051	3,28
34	3	95,6	0,125	4,7	95,8	0,0824	4,62
35	4	99,6	0,0468	4,96	99,5	0,0551	4,82
36	5	99,6	0,052	4,93	99,6	0,0539	5,18

Tabla E.10: Valores para P al variar D



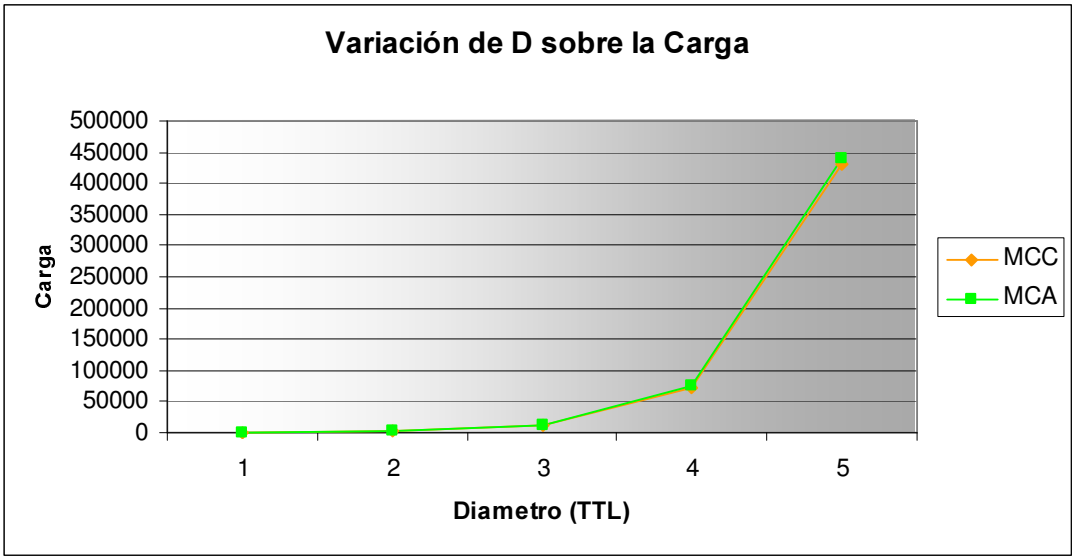
Grafica E.8: Comportamiento de P al variar D

Para el caso del porcentaje de nodos conectados por caminos de largo menor o igual a D (P), el comportamiento esperado es también un aumento al aumentar D, por las mismas razones esgrimidas en el caso anterior: al aumentar el largo permitido de los caminos aumenta la probabilidad de que desde cada nodo se puedan alcanzar nuevos nodos, lo que en definitiva aumenta la cantidad de pares de nodos conectados y, por ende, el porcentaje de los mismos sobre la cantidad total de pares de nodos del grafo.
Esta idea se ve respaldada por los resultados obtenidos, tanto mediante MCC como por MCA, observándose, al igual que en al caso de la confiabildad, un aumento rápido en el valor de la función al variar D entre 1 y 3, hasta que se alcanza el valor umbral P=100, a partir del cual lógicamente el mismo se mantiene constante. El corrimiento de los valores de D (antes era entre 3 y 5) en los cuales se produce el rápido aumento se debe a que la función de porcentaje es menos ”exigente” que la de CDA.

Parámetros fijos: N = 50, CCP = 6, CPN = 1, M = 500

Caso	D	Carga						
		MCC			MCA			
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)	Formula
32	1	301	1,02	0,031	300	0,154	0,143	299
33	2	2063	12,8	0,281	2069	2,65	0,451	2063
34	3	12440	124	2,52	12372	36,5	2,38	12426
35	4	73784	934	14,3	73820	336	14	73340
36	5	430856	7492	95,5	439492	3364	99,4	431359

Tabla E.11: Valores para la carga al variar D



Grafica E.9: Comportamiento de C al variar D

Por último veremos el comportamiento de la función de carga ante la variación de D. En la gráfica se nota un aumento paulatino, que podría ser exponencial o polinominal, de la misma al ir aumentando el valor de D. Este comportamiento respalda de alguna forma, la fórmula de carga que se definió como estimador de la cantidad de paquetes enviados por los nodos de la red, la cual indicaba precisamente un crecimiento de tipo exponencial en la cantidad de paquetes respecto a D.
Este hecho refleja el comportamiento esperado en nuestro modelo, ya un aumento de D en definitiva refleja un aumento en el tiempo de vida (TTL) de los paquetes en la red, aumentando consiguientemente la carga en la misma.

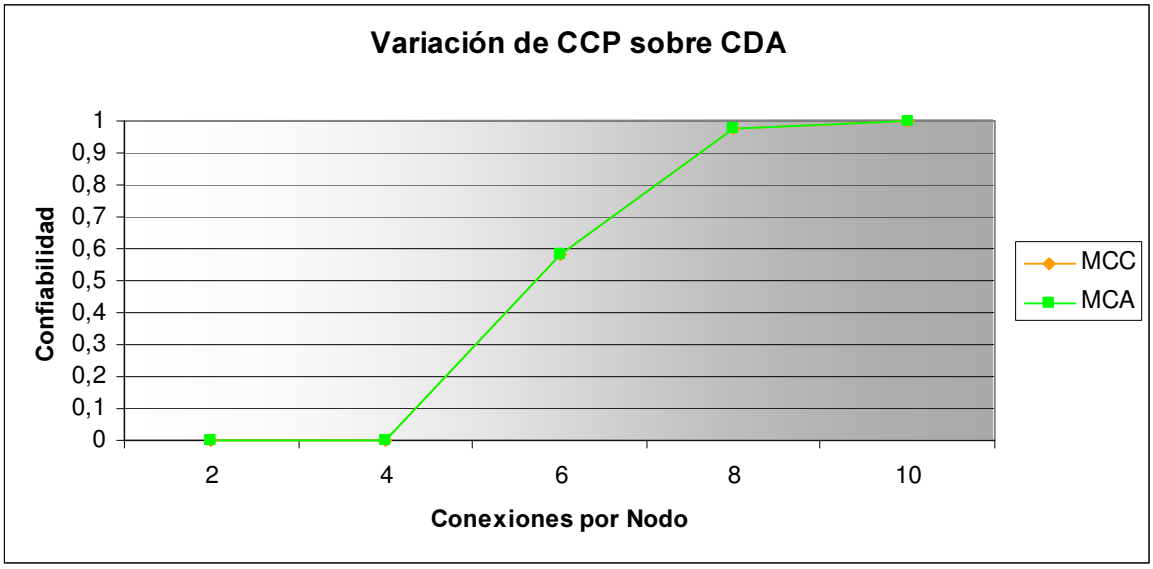
E.5.3.3.2. Variación de CCP.

Como segunda prueba, analizaremos el comportamiento de las funciones de evaluación al variar la cantidad de conexiones promedio por nodo (CCP) permitidas, manteniendo fijos los demás parámetros de entrada del sistema.

Parámetros fijos: N = 50, D = 4, CPN = 1, M = 500

		Diámetro Acotada					
Caso	CCP	MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	Tiempo
37	2	0	0	0	0	0	0,016
38	4	0	0	0,141	0	0	0,187
35	6	0,584	0,022	3,11	0,584	0,0248	2,74
39	8	0,974	0,00712	5,07	0,977	0,00583	4,84
40	10	1	0	6,57	1	0	6,82

Tabla E.12: Valores de CDA al variar CCP



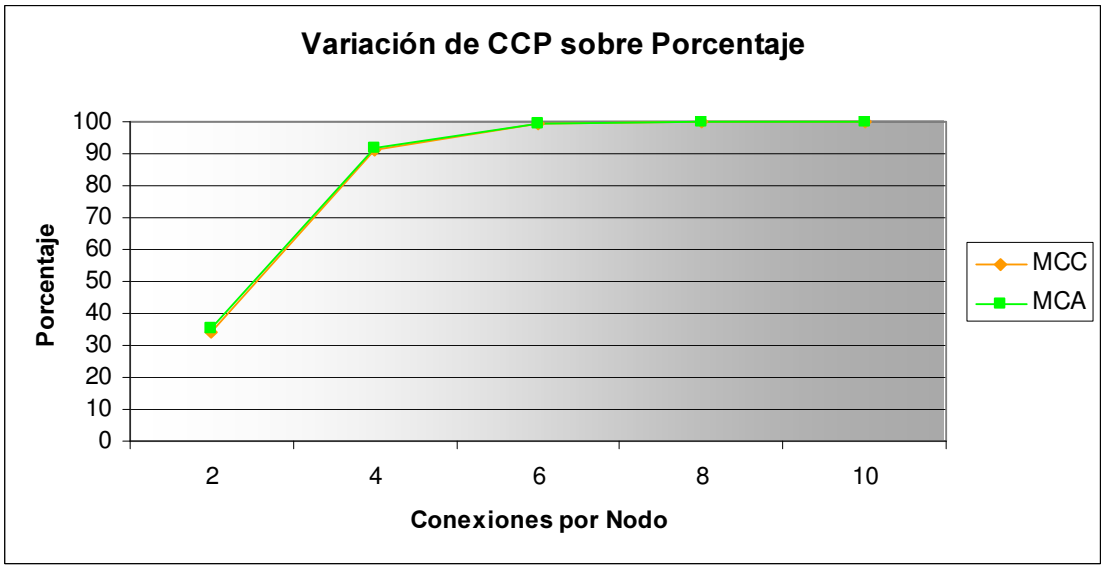
Grafica E.10: Comportamiento de CDA al variar CCP

Un aumento en la cantidad de conexiones por nodo debería ocasionar un aumento en la CDA, debido al incremento en la amplitud de la búsqueda de la información en cada nivel de la misma. Esta idea se confirma en los resultados obtenidos, notándose un incremento abrupto en un punto, de forma similar a lo descrito para la confiabilidad al variar D (Gráfica E.7)

Parámetros fijos: N = 50, D = 4, CPN = 1, M = 500

Caso	CCP	Porcentaje					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	Tiempo
37	2	34	0,464	1,31	35,5	0,265	1,37
38	4	91,3	0,248	3,42	91,7	0,161	3,44
35	6	99,6	0,0468	4,96	99,5	0,0551	4,82
39	8	99,9	0,0224	5,72	99,9	0,0112	5,79
40	10	100	0	7,17	100	0	7,1

Tabla E.13: Valores de P al variar CCP



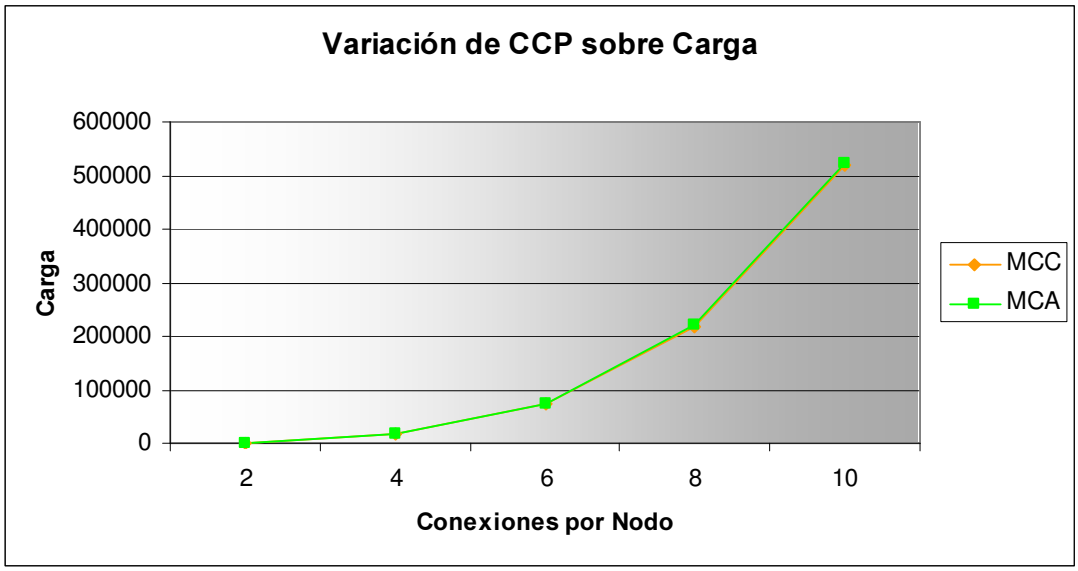
Grafica E.11: Comportamiento de P al variar CCP

De forma análoga, es de esperarse un aumento en el porcentaje de pares de nodos conectados al aumentar la CCP, comportamiento que se puede visualizar en la Gráfica E.11, que muestra la variación de P al cambiar el valor de CCP.
Este aumento también tiene saltos importantes, entre los valores de CCP = 2 y CCP = 4, pero luego continúa con un aumento paulatino hasta alcanzar el valor máximo P = 100 en donde se mantiene constante.

Parámetros fijos: N = 50, D = 4, CPN = 1, M = 500

Caso	CCP	Carga						
		MCC			MCA			
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)	Formula
37	2	1396	31,9	0,752	1460	19,9	0,623	1431
38	4	16078	265	4,41	16085	100	3,89	16086
35	6	73784	934	14,3	73820	336	14	73340
39	8	215999	2318	36,7	219985	875	37,5	220615
40	10	518399	5487	85	521272	2918	86	523386

Tabla E.14: Valores de la carga al variar CCP



Grafica E.12: Comportamiento de C al variar CCP

Cerrando el análisis del comportamiento de las funciones al variar CCP, solo nos resta ver la función de carga. A simple vista, la misma aparenta tener un comportamiento polinominal, lo que, al igual que en el caso en que variamos D, refuerza el respaldo a la fórmula de carga definida.
Este comportamiento era de esperarse, ya que, como habíamos mencionado, un aumento de CCP en definitiva genera un aumento en la amplitud de la búsqueda que se acumula en cada nuevo nivel de la misma y de ahí su comportamiento polinominal en CCP.

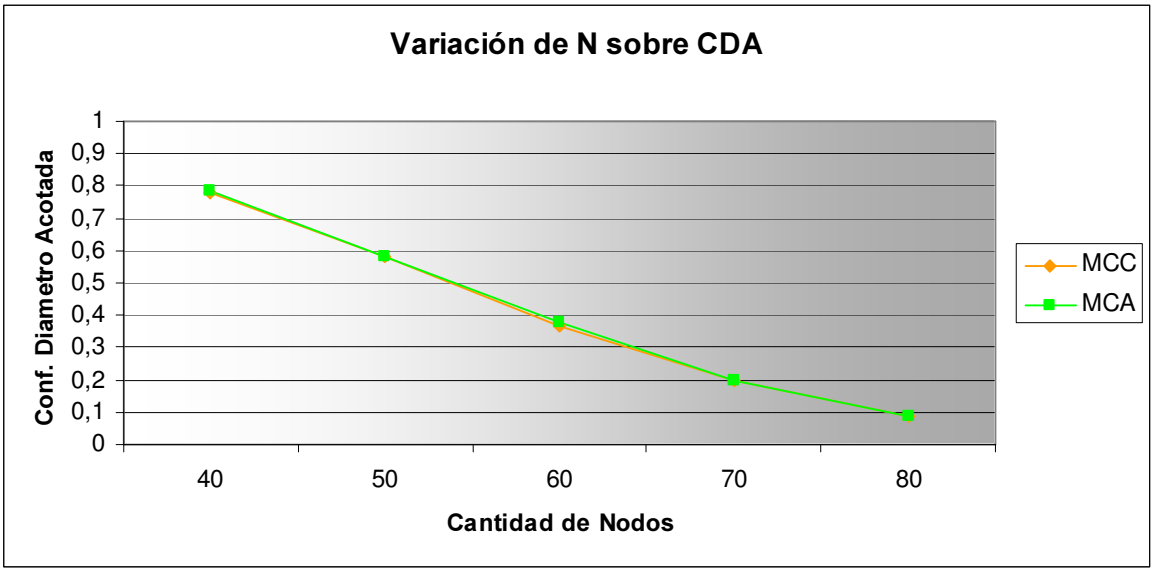
E.5.3.3.3. Variación de N

Para finalizar con esta sección del análisis, presentaremos los resultados y los comentarios correspondientes para las tres funciones de interés sobre los grafos al variar la cantidad de nodos (N)

Parámetros fijos: CCP = 6, D = 4, CPN = 1, M = 500

Caso	N	Diámetro Acotada					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)
41	40	0,778	0,0186	2,38	0,784	0,0199	2,25
35	50	0,584	0,022	3,11	0,584	0,0248	2,74
42	60	0,368	0,0215	3,13	0,376	0,0188	3,17
43	70	0,198	0,0178	2,8	0,198	0,0162	2,98
44	80	0,088	0,0126	2,69	0,085	0,00956	2,24

Tabla E.15: Valores de CDA al variar N



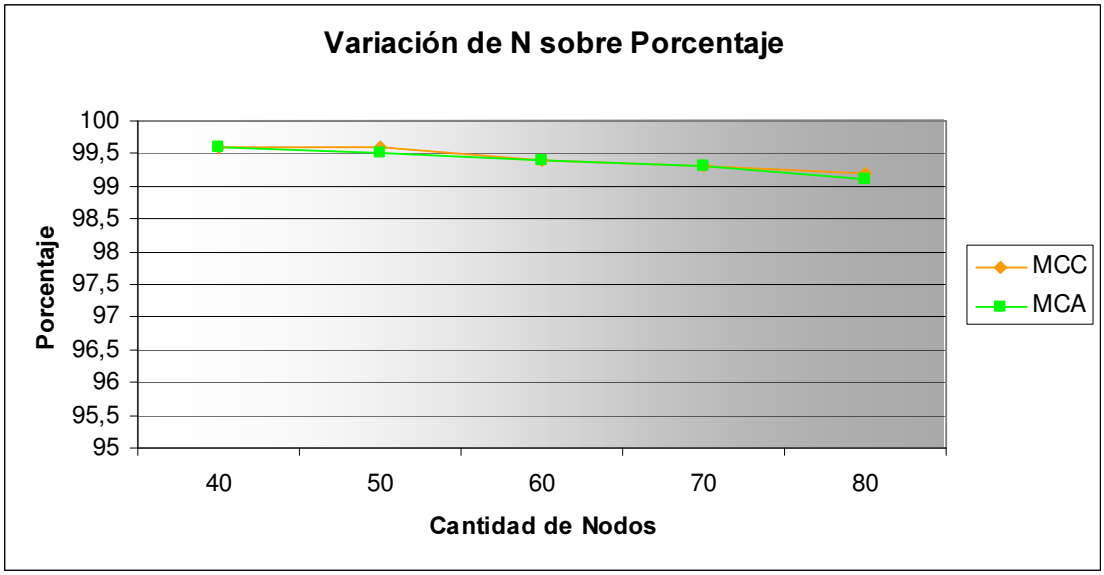
Grafica E.13: Comportamiento de CDA al variar N

A diferencia de lo que eran los parámetros que analizamos en los dos puntos anteriores, un aumento en la cantidad de nodos del grafo, manteniendo fijos los demás parámetros, produce una disminución de la CDA. Esto es de esperarse, ya que un aumento en el número de nodos implica en todos los nodos del grafo un aumento de los destinos a los que se debe llegar sin posibilidad de extender la búsqueda de los mismos tanto en amplitud como en profundidad, lo que genera la mencionada caída en la probabilidad de que todos los nodos de la red puedan llegar a todos los demás a través de camino de a lo sumo largo D.

Parámetros fijos: CCP = 6, D = 4, CPN = 1, M = 500

Caso	N	Porcentaje					
		MCC			MCA		
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)
41	40	99,6	0,0542	2,74	99,6	0,0478	3,17
35	50	99,6	0,0468	4,96	99,5	0,0551	4,82
42	60	99,4	0,054	7,24	99,4	0,0405	6,57
43	70	99,3	0,0496	9,88	99,3	0,0556	10
44	80	99,2	0,0464	14,4	99,1	0,0448	13,8

Tabla E.16: Valores de P al variar N



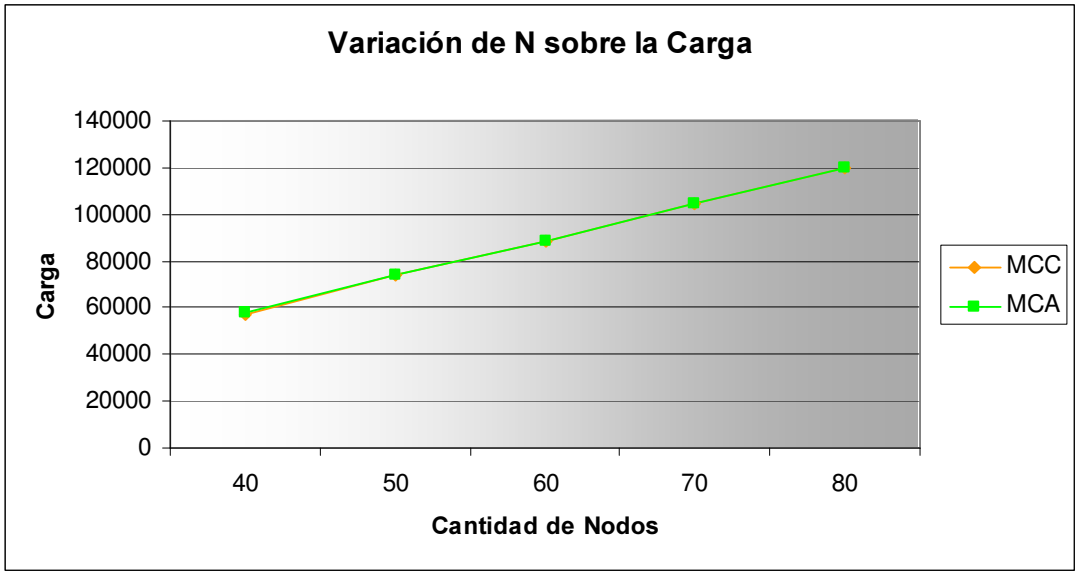
Grafica E.14: Comportamiento de P al variar N

Para la función que mide el porcentaje de nodos conectados, un comportamiento similar al de la función de confiabilidad es observado. La misma disminuye paulatinamente al aumentar N, aunque esta disminución es de carácter más gradual que en el caso anterior.
En un análisis inicial, las causas de este comportamiento son similares a las planteadas para el caso de la función de confiabilidad: el agregado de nuevos nodos al grafo sin variar el resto de los parámetros incide en una disminución del número de nodos que podemos alcanzar desde un nodo particular. Esto incide en el aumento del numerador del cociente en la fórmula del porcentaje, pero si miramos su denominador (la cantidad de pares de nodos del grafo) éste también aumenta al crecer N, lo que de alguna forma lleva a una disminución "más lenta" de la función de porcentaje respecto a la de confiabilidad diámetro acotada.

Parámetros fijos: CCP = 6, D = 4, CPN = 1, M = 500

Caso	N	Carga						
		MCC			MCA			
		Valor	Sigma	T (seg.)	Valor	Sigma	T (seg.)	Formula
41	40	57223	794	11	57908	336	10,6	577799
35	50	73784	934	14,3	73820	336	14	73340
42	60	88483	1066	18,1	88652	660	17	88880
43	70	104551	1269	23	104836	549	23,2	104420
44	80	119978	1265	29,4	119933	483	29,4	119961

Tabla E.17: Valores de la carga al variar N



Grafica E.15: Comportamiento de C al variar N

A diferencia de las dos funciones anteriores en este estudio del comportamiento al variar N, la función de carga presenta un comportamiento creciente que, a simple vista, aparenta ser lineal. Y que nuevamente respalda la fórmula de carga planteada.

Para los tres puntos que se acaban de desarrollar referentes al comportamiento de las funciones de evaluación al fijar un parámetro y variar el resto, cabe extender el comentario hecho al comienzo del presente documento: todos los comentarios realizados se basan en los resultados obtenidos en el caso de estudio particular y formalmente solo son válidos en ese contexto, pero tomamos los mismos de base para una estimación de cómo es el comportamiento en el conjunto de los casos.

E.5.4. Casos de borde y casos no validos

Una parte no despreciable de la validación en cualquier desarrollo de software, es la prueba de casos determinados de antemano como no válidos para el sistema y de los casos de borde del mismo, tarea que encaramos en la presente sección.

Empezaremos por definir informalmente qué es en nuestro contexto de trabajo un caso no válido y un caso de borde:

- Caso no válido: conjunto de parámetros del algoritmo que rompe alguna de las propiedades con las que se definió el modelo en que se encuadra el mismo.

- Caso de borde: conjunto de parámetros que siendo válido según la definición modelo, la variación en una unidad de alguno de sus parámetros puede convertirlo en un caso no válido.

Un ejemplo de un caso no válido sería cualquier conjunto de parámetros cuya cantidad de nodos sea menor que 2, ya que el modelo define los grafos como formados por un conjunto de al menos dos nodos. Por otro lado, un conjunto de parámetros cuya cantidad de nodos sea igual a 2 sería un ejemplo de caso de borde.

Los casos no válidos que se definieron y el resultado que se obtuvo para cada uno, son los siguientes:

Caso	Descripción	Resultado
22	Cantidad de Nodos <= 1	Ok
23	Cantidad de Conexiones >= Cantidad de Nodos	Ok
24	Cualquier parámetro con un valor negativo (exceptuando la semilla)	Ok
25	Tamaño de la muestra <= 1	Ok

Tabla E.18: Casos no válidos

En la columna '*Resultado*', con OK nos referimos a que el sistema controla el caso ingresado y despliega una ventana de mensaje informando la situación correctamente.

Seguidamente presentamos los casos de borde que se detectaron también acompañados de los resultados que se obtuvieron para cada uno. Dado que en estos casos de prueba el sistema si ejecuta los algoritmos, tenemos resultados que podemos analizar, cosa que hacemos después de la tabla:

Caso		Entrada				CDA			Porcentaje			Carga		
	Descripción	N	CCP	D	CPN	GCE	MCC	MCA	GCE	MCC	MCA	GCE	MCC	MCA
16	Conexiones = Cant. Nodos -1	5	4	1	1	1	1	1	100	100	100	20	20	20
17	Cant. Nodos = 2	2	0	1	1	0	0	0	0	0	0	0	0	0
18	Diámetro = 0	5	3	0	1	0	0	0	0	0	0	0	0	0
19	Conexiones = 0	4	0	2	1	0	0	0	0	0	0	0	0	0
20	Pedidos = 0	5	3	2	0	1	1	1	100	100	100	0	0	0
21	Diametro = 1 con Conexiones != Cant. Ndos -1	5	3	1	1	0,06	0,064	0,0666	N/A	N/A	N/A	N/A	N/A	N/A

Tabla E.19: Casos de borde

A continuación comentaremos los resultados obtenidos en estos casos de borde con mayor detalle:

- **Caso 16**: CCP = N -1 corresponde al caso de la confiabilidad clásica, entonces CDA = CCP/N-1 = 1.
- **Caso 17**: Para N = 2 tenemos dos alternativas: si CCP = 1 entonces CDA = 1, P = 100 y C = 1; y si CCP = 0 entonces CDA = 0, P = 0 y C = 0, que en definitiva fue el caso que se probó con resultados satisfactorios.
- **Caso 18**: Definiendo el parámetro D = 0, es de esperarse que todas las funciones retornen 0, ya que no es posible el establecimiento de conexiones entre los nodos ni consecuentemente el envío de paquetes.
- **Caso 19**: Una situación similar a la anterior tenemos al definir la cantidad de conexiones permitidas a cada nodo en promedio CCP = 0, ya que al no permitir conexiones tampoco es posible el envío de paquetes y por consiguiente todas las funciones de evaluación deberían retornar, como efectivamente sucede.
- **Caso 20**: Este caso es de interés para el análisis de la función de carga, ya que el hecho de que ningún nodo envíe un paquete no afecta los valores de las funciones de CDA y de porcentaje. Para la carga parece intuitivo pensar que si no se envían paquetes la misma sea cero, cosa que se comprueba empíricamente para el caso de prueba elegido.
- **Caso 21**: Cuando el diámetro máximo para que la consideración de un camino como operacional (D) se define en 1, sucede que las únicas instancias de grafos en los que la CDA es 1 son aquellas en las que el grafo es completo, de forma que podemos calcular de manera sencilla la CDA para ese caso. La misma está dada por la probabilidad de que el sorteo para cada arista sea positivo (o sea, que la arista se encuentre activa), o sea, por la probabilidad de que una arista esté operativa elevado a la cantidad de aristas del grafo completo:

  ➔ CDA = CCP/N-1 ^ (N*(N-1)/2)

  Este cálculo para los parámetros utilizados en el caso de prueba nos da un valor para CDA de 0,0563 que coincide con el valor obtenido producto de la ejecución de los algoritmos.

Dado que en esta prueba en particular no revisten de mayor interés los resultados de las demás funciones, las columnas correspondientes a los resultados de las mismas se han completado con “N/A” (No Aplica).

E.5.5. Comparación MCC vs. MCA

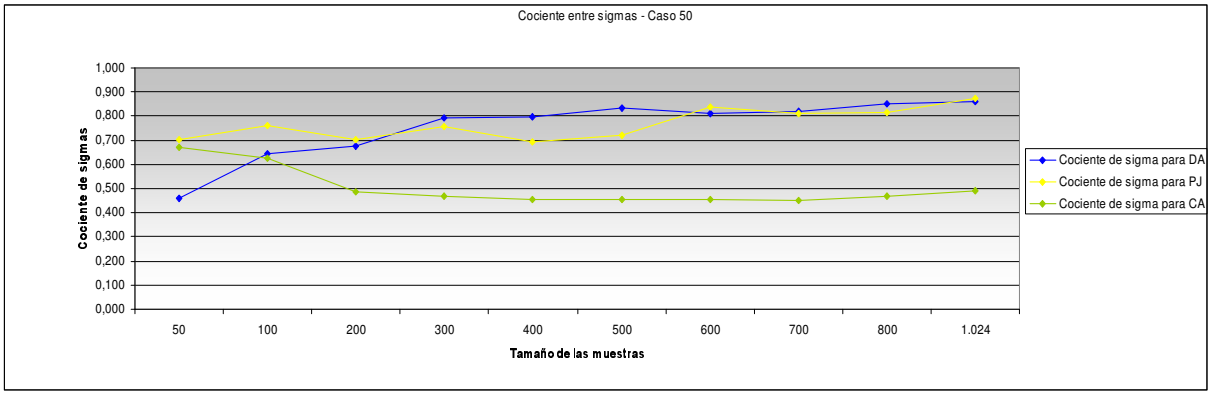
Otro análisis que realizamos de los resultados obtenidos de las pruebas de los algoritmos de evaluación es el correspondiente a la comparación de las dos variantes del método de Monte Carlo implementadas: el Monte Carlo Crudo y el Monte Carlo Antitético.
Teóricamente el MCA debería provocar una reducción de la varianza en los resultados, hecho que intentaremos comprobar empíricamente. El enfoque que seguiremos es el de tomar cuatro casos arbitrarios y estudiar el cociente de las desviaciones estándar (Sigma(MCC) / Sigma(MCA)) de los resultados obtenidos por cada uno de los métodos. El estudio de estos cocientes se hará para las tres funciones de evaluación.

Los juegos de datos en el que se basan las gráficas que se presentan son los siguientes:

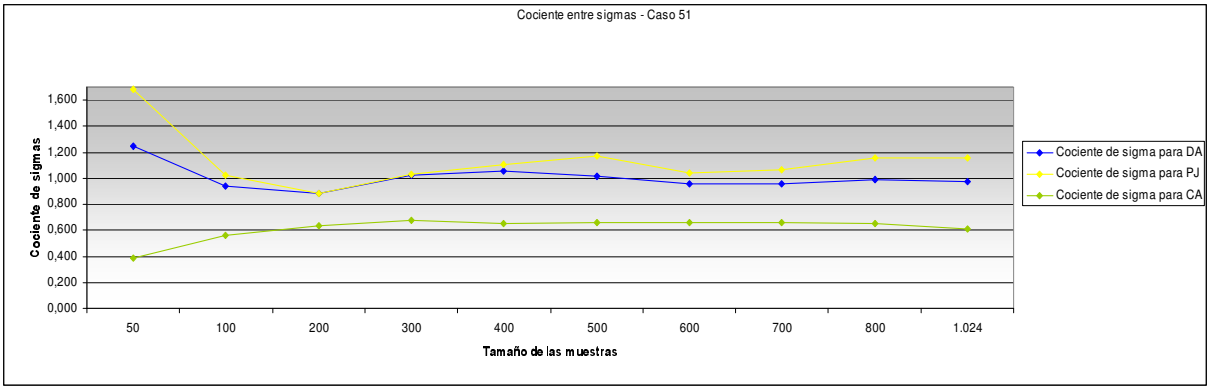
Caso	N	CCP	D	CPN
50	30	6	3	1
51	50	20	2	4
52	70	22	2	2
53	100	12	3	1

Tabla E.20: Juegos de datos para comparación MCC vs. MCA

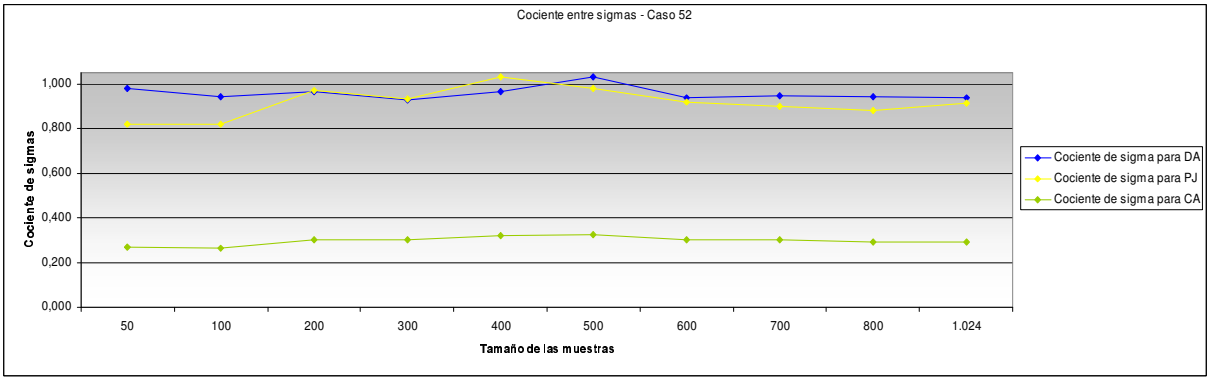
A continuación presentamos las gráficas obtenidas resultados de los casos de prueba ejecutados.



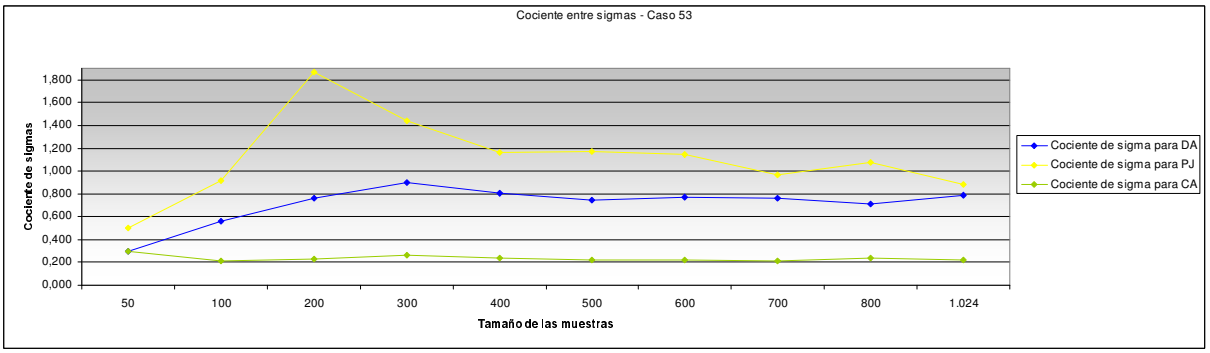
Grafica E.16: Estudio de desviaciones estándar del MCC y el MCA – Caso 50



Grafica E.17: Estudio de desviaciones estándar del MCC y el MCA – Caso 51



Grafica E.18: Estudio de desviaciones estándar del MCC y el MCA – Caso 52



Grafica E.19: Estudio de desviaciones estándar del MCC y el MCA – Caso 53

Para los casos estudiados el comportamiento de las desviaciones estándar en ambos métodos, no muestra una tendencia clara en cuanto a la reducción esperada para las mismas, salvo en el caso de la función de carga, en donde la reducción de varianza es notoria.
Este comportamiento puede tener su causa en que no estamos calculando la varianza real del MCA (que está probado teóricamente que es menor que la varianza del MCC) sino un estimador de la misma, de cuya calidad dependen los resultados que se obtienen.

Analizando el comportamiento de estas funciones cociente precisamente al variar la cantidad de muestras que se toman, es apreciable una tendencia a la estabilización del mismo (y presumiblemente de las varianzas de cada uno de los métodos) ante el aumento de la cantidad de muestras.

Una particularidad que se observa es que el cociente de las desviaciones estándar de la función de porcentaje es el que indica un comportamiento más dispar entre los dos métodos. Esto se podría deber a que precisamente esta función es la que tiende a dar valores más cercanos a 1 (100) y en estos casos es donde el MCC presenta las mayores debilidades en sus resultados en el sentido de retornar valores con altas varianzas.

En concreto, el comportamiento de reducción de varianza esperado para el MCA respecto al MCC fue, de alguna forma, comprobado empíricamente con las pruebas realizadas.

Por otro lado, además del análisis referente a los resultados obtenidos mediante cada uno de los métodos Monte Carlo, también se realizó un estudio de performance de las dos implementaciones presentadas.
El mismo se resume en las siguientes tres tablas en donde presentamos consecutivamente los resultados obtenidos al tomar un caso base y variar luego cada uno de los tres parámetros de interés del problema.

Caso base: N = 50, CCP = 6, CPN = 1, M = 500

Caso	D	Diámetro Acotada		Porcentaje		Carga		Tiempos (seg.)			
		MCC	MCA	MCC	MCA	MCC	MCA	MCC		MCA	
		T (seg.)	T (seg.)	T (seg.)	T (seg.)	T (seg.)	T (seg.)	T.Total	Gener.	T.Total	T. Gener.
32	1	0	0,047	1,2	1,03	0,031	0,143	1,56	21,1%	26,6	95,4%
33	2	0,112	0,93	3,42	3,28	0,281	0,451	3,98	4,2%	28,7	83,8%
34	3	0,203	0,25	4,7	4,62	2,52	2,38	7,85	5,4%	34	78,7%
35	4	3,11	2,74	4,96	4,82	14,3	14	22,6	1,0%	48,6	55,6%
36	5	4,36	4	4,93	5,18	95,5	99,4	105	0,2%	136	20,2%

Tabla E.21: Análisis de performance MCC vs. MCA variando D

Caso base: N = 50, D = 4, CPN = 1, M = 500

Caso	CCP	Diámetro Acotada		Porcentaje		Carga		Tiempos (seg.)			
		MCC	MCA	MCC	MCA	MCC	MCA	MCC		MCA	
		T (seg.)	Tiempo	T (seg.)	Tiempo	T (seg.)	T (seg.)	T.Total	Gener.	T.Total	Gener.
37	2	0	0,016	1,31	1,37	0,752	0,623	2,53	18,5%	29,1	93,1%
38	4	0,141	0,187	3,42	3,44	4,41	3,89	8,32	4,2%	31,1	75,8%
35	6	3,11	2,74	4,96	4,82	14,3	14	22,6	1,0%	48,6	55,6%
39	8	5,07	4,84	5,72	5,79	36,7	37,5	47,8	0,6%	73,6	34,6%
40	10	6,57	6,82	7,17	7,1	85	86	99,2	0,5%	125	20,1%

Tabla E.22: Análisis de performance MCC vs. MCA variando CCP

Caso base: CCP = 6, D = 4, CPN = 1, M = 500

Caso	N	Diámetro Acotada		Porcentaje		Carga		Tiempos (seg.)			
		MCC	MCA	MCC	MCA	MCC	MCA	MCC		MCA	
		T (seg.)	T (seg.)	T (seg.)	T (seg.)	T (seg.)	T (seg.)	T.Total	Gener.	T.Total	Gener.
41	40	2,38	2,25	2,74	3,17	11	10,6	16,3	1,1%	26,4	39,3%
35	50	3,11	2,74	4,96	4,82	14,3	14	22,6	1,0%	48,6	55,6%
42	60	3,13	3,17	7,24	6,57	18,1	17	28,8	1,1%	81,9	67,4%
43	70	2,8	2,98	9,88	10	23	23,2	36	0,9%	145	75,0%
44	80	2,69	2,24	14,4	13,8	29,4	29,4	47	1,1%	231	80,3%

Tabla E.23: Análisis de performance MCC vs. MCA variando N

Las tres tablas siguen el mismo formato: en las columnas '*Diámetro Acotada*', '*Porcentaje*' y '*Carga*' presentan los tiempos registrados para las funciones de confiabilidad diámetro acotada, porcentaje de nodos conectados por caminos de largo menor o igual a *D*, y carga respectivamente. Las últimas cuatro columnas corresponden a los tiempos totales obtenidos para la ejecución de las tres funciones con cada uno de los métodos. Como se verá éste valor no coincide con la suma de los tiempos obtenidos para las tres funciones, correspondiendo la diferencia al tiempo en que el algoritmo se encuentra generando los estados para las evaluaciones. Las columnas '*Gener.*' Indican precisamente el porcentaje del tiempo total de

ejecución de cada método en que el algoritmo estuvo generando los estados para la evaluación.
En todos los casos los tiempos expresados se encuentran en segundos.

Entrando en el análisis de los resultados obtenidos, es notoria la mejor performance observada para el MCC frente al MCA, por lo menos en los casos estudiados. También salta a la vista el hecho de que para el MCA un alto porcentaje del tiempo de ejecución se consume en la generación de los estados, lo que introduce una incertidumbre en la comparación de los tiempos de ejecución en sí de los algoritmos que se pretenden comparar.
Se nota también un descenso paulatino del porcentaje de tiempo en la generación de los estados en ambos métodos, al aumentar los valores de *D* y *CCP*, manteniendo el resto de los parámetros fijos. Este comportamiento no se repite en la serie de casos en que aumentamos *N* para un mismo caso base.

E.5.6. Análisis de resultados obtenidos: Java vs. C++

Cerrando los que es la validación y análisis de resultados de la herramienta de evaluación, presentaremos la comparación de resultados y performance de los métodos implementados en Java contra los implementados en C++.
El enfoque planteado en esta etapa de las pruebas fue el de elegir un caso base, evaluarlo, y luego ir tomando cada uno de los parámetros y, manteniendo fijos el resto, realizar dos corridas para diferentes valores del parámetro en cuestión. De esta forma haciendo dos corridas para cada parámetro, podemos tener un conjunto de tres corridas de donde poder armar una mínima idea de cómo afecta la variación de este parámetro a los resultados en cada uno de los lenguajes.

Vale aclarar de antemano una situación detectada mientras se hacían algunas de estas pruebas sobre los algoritmos en C++. En un momento se notó que la gran mayoría de los resultados eran mayores que los obtenidos en Java, e investigando se llegó a la conclusión de que esto se debía a un sesgo en el generador de números aleatorios de C++. Pruebas realizadas ejecutando los algoritmos en C++ alimentados por números aleatorios obtenidos mediante el generador de Java, comprobaron éste hecho y nos sirvieron para, de alguna forma, probar la correctitud de los algoritmos implementados en C++ ya que los resultados coincidieron exactamente con los que se obtuvieron para Java.

En definitiva, los resultados obtenidos en esta prueba de variación individual de un parámetro son los que se presentan en la siguiente tabla:

Caso	Variando	N	CCP	D	CPN	M	Fi	Java			C++			T_{Java} / T_{C++}
								Val	Sig	T(seg.)	Val	Sig	T(seg.)	
45	Caso de ref.	1000	20	5	1	10	DA	1	0	180,00	1	0	164,00	1,098
46	C	1000	10	5	1	10	DA	0,9	0,999	124,00	0,8	0,133	118,00	1,051
47	C	1000	50	5	1	10	DA	1	0	332,00	1	0	305,00	1,089
48	D	1000	20	3	1	10	DA	0	0	0,33	0	0	0,30	1,097
49	D	1000	20	10	1	10	DA	1	0	177,00	1	0	163,00	1,086
54	N	500	20	5	1	10	DA	1	0	34,50	1	0	31,70	1,088
55	N	1500	20	5	1	10	DA	1	0	493,00	1	0	477,00	1,034
56	Fi	1000	20	5	1	10	PO	100	0	194,00	100	0	165,00	1,176
57	Fi	1000	20	5	1	10	CA	3,2552E+09	5,0118E+07	20741,59	4,4413E+09	7,7817E+07	29514,00	0,703

Tabla E.24: Comportamiento al variar individualmente los parámetros en C++ y Java

Por tratarse de un lenguaje que se ejecuta compilado (C++) contra otro que ejecuta interpretado (Java) se puede esperar una mejor performance en cuanto a tiempo de ejecución en los métodos implementados en C++.
Las conclusiones que podamos sacar respecto a este tema con esta prueba específica son bastante parciales, ya en la mayoría de los casos la función elegida para la ejecución es la CDA. Con respecto a ésta función observamos que efectivamente el tiempo de ejecución en

C++ es menor al de Java, pero solo en el orden del 10%. Por otro lado observando la única ejecución de la función de carga se observa el comportamiento contrario, situación que podremos estudiar con más detalles en la siguiente prueba.

Justamente esta segunda prueba de análisis de los resultados obtenidos en los dos lenguajes elegidos se basa en la selección de un grupo arbitrario de casos y en su evaluación en C++ y Java mediante los dos de los tres métodos implementados (MCA no fue implementado en C++ por lo que no se pueden análisis referentes a éste), teniendo en mente tanto los resultados obtenidos como los tiempos de ejecución registrados.

Entrada							CDA			Porcentaje			Carga		
Caso	Leng.	N	CCP	D	CPN	M	GCE	MCC		GCE	MCC		GCE	MCC	
							Valor	Limite inf.	Limite sup.	Valor	Limite inf.	Limite sup.	Valor	Limite inf.	Limite sup.
14	Java	7	4	2	1	20971	0,807	0,801	0,817	98,2	98,10	98,30	121	121	122
14	C++	7	4	2	1	20971	0,807	0,809	0,825	98,2	98,21	98,39	121	122	122
10	Java	6	3	2	2	16384	0,603	0,590	0,612	93,2	93,03	93,57	122	121	123
10	C++	6	3	2	2	16384	0,603	0,601	0,623	93,2	93,13	93,67	122	122	124
29	Java	8	3,15	7	1	1000	N/A	0,850	0,912	N/A	95,58	97,42	N/A	16.802	24.758
10	C++	8	3,15	7	1	1000	N/A	0,860	0,920	N/A	95,89	97,71	N/A	18.886	26.320
31	Java	10	4,05	9	1	500	N/A	0,908	0,972	N/A	97,89	99,31	N/A	23.314.881	24.573.675
31	C++	10	4,05	9	1	500	N/A	0,927	0,983	N/A	98,20	99,60	N/A	1.999.582	3.395.542
35	Java	50	6	4	1	500	N/A	0,518	0,650	N/A	99,46	99,74	N/A	70.982	76.586
35	C++	50	6	4	1	500	N/A	0,542	0,672	N/A	99,33	99,67	N/A	71.867	78.191
42	Java	60	6	4	1	500	N/A	0,304	0,433	N/A	99,24	99,56	N/A	85.285	91.681
42	C++	60	6	4	1	500	N/A	0,326	0,456	N/A	99,25	99,55	N/A	89.193	96.369
41	Java	40	6	4	1	500	N/A	0,722	0,834	N/A	99,44	99,76	N/A	54.841	59.605
41	C++	40	6	4	1	500	N/A	0,755	0,861	N/A	99,44	99,76	N/A	57.246	62.490
34	Java	50	6	3	1	500	N/A	0,000	0,008	N/A	95,23	95,98	N/A	12.068	12.812
34	C++	50	6	3	1	500	N/A	0,000	0,000	N/A	95,53	96,27	N/A	12.499	13.213

Tabla E.25: Comparación de ejecuciones en C++ y Java – Resultados

En todos los casos estudiados los resultados son satisfactorios en el sentido de para la GCE (en los casos que es posible de ejecutar) los resultados en C++ y Java coinciden, mientras que para el MCC los intervalos se intersectan, salvo en el caso **31** para la función de carga, en donde los resultados son completamente dispares.
Se confirma aquí también el comportamiento mencionado al comienzo de este capítulo de que los resultados en C++ tienden a ser mayores que los de Java, hecho que es producto del generador de números aleatorios que brinda el lenguaje C++.

Entrada							Tiempo (seg.)			
Caso	Leng.	N	CCP	D	CPN	M	CDA	Porc.	Carga	Total
14	Java	7	4	2	1	20971	101	195	82,9	404
14	C++	7	4	2	1	20971	204	391	212	821
10	Java	6	3	2	2	16384	1,22	1,89	1,2	4,57
10	C++	6	3	2	2	16384	2,7	4,5	2,32	9,64
29	Java	8	3,15	7	1	1000	0,158	0,111	11,4	11,8
10	C++	8	3,15	7	1	1000	0,203	0,186	29,6	30
31	Java	10	4,05	9	1	500	0,1239	0,252	636	636
31	C++	10	4,05	9	1	500	0,124	0,188	1596	1596
35	Java	50	6	4	1	500	3,1	4,25	12,7	20,4
35	C++	50	6	4	1	500	4	5,33	24,7	34,7
42	Java	60	6	4	1	500	3,03	6,48	16,3	25,81
42	C++	60	6	4	1	500	4,1	8,31	31,1	43,51
41	Java	40	6	4	1	500	2,37	2,54	9,73	14,9
41	C++	40	6	4	1	500	3,09	3,05	19,6	25,9
34	Java	50	6	3	1	500	0,283	3,82	2,02	6,37
34	C++	50	6	3	1	500	0,277	5,58	4,12	10

Tabla E.26: Comparación de ejecuciones en C++ y Java – Tiempos

En lo que a tiempos de ejecución respecta, el comportamiento es de alguna forma contradictorio, respecto al observado en las primeras pruebas (Tabla 5.9) y al esperado, ya que las corridas en C++ muestran una menor performance que las de Java.
En un primer análisis este comportamiento no tiene un origen claro. Queda para algún trabajo futuro la tarea de hacer un análisis más detallado del mismo en busca de sus causantes.

E.6. Trabajo futuro

E.6.1. Extensiones

Algunas extensiones que surgen naturalmente al momento de pensar en mejoras para la aplicación son las referentes a agregar nuevas funciones de interés a evaluar. Del diseño planteado para la aplicación se desprende que una extensión de éste tipo no sería muy compleja, ya que en principio bastaría con agregar la clase que implemente esta nueva función como una subclase de *Fi* y efectuar algunos arreglos menores en algunas otras clases, básicamente en las que manejan la interfaz gráfica de la aplicación.
De la misma forma se podría pensar en la implementación de otros métodos para la evaluación de las diferentes funciones, lo que se traduciría en un cambio similar al planteado arriba, solo que en este caso la nueva clase se agregaría como subclase de *Metodo*.

Otras extensiones que se pueden pensar para la aplicación serían:

- Posibilidad de tomar la entrada desde un archivo (XML o de texto) que contendría un conjunto de juegos de parámetros con los que se iría alimentando el motor de cálculo, retornándose luego los resultados obtenidos para cada una pruebas en un archivo de salida. Esta idea sería de utilidad para la realización de un número importante de pruebas de manera sencilla.
  Siguiendo este enfoque, se podría pensar también en un motor de análisis automático de los resultados que se obtenga, con la posibilidad de generar salidas en forma de gráfica para el conjunto de juegos de parámetros ingresado.

- Extender la interfaz gráfica para permitir el manejo de grafos no completos, por ej., tomando la información de la topología del grafo y las confiabilidades de las aristas de un archivo GXL. La ampliación en este sentido sería solo a nivel de la interfaz gráfica de la aplicación ya que el motor de evaluación de la misma ya está implementado considerando grafos genéricos.

E.6.2. Bugs

En las pruebas realizadas no se detectaron problemas con la aplicación.

E.7. Referencias

[CEK 94]
A recursive variance-reduction algorithm for estimating communication-network reliability
H. Cancela, M. El Khadiri
Technical Report PI860, IRISA, Rennes, Setiembre de1994.

[Coul 87]
The Combinatorics of Network Reliability.
C. Colbourn.
Oxford University Press, New York - 1998

[CRU 01]
An algorithm to compute the all-terminal reliability measure.
H. Cancela, G. Rubino, M. E. Urquhart.
Journal of the Indian Operational Research Society (OPSEARCH), Vol. 38, No.6, pp. 567-579, 2001.

[CU 93]
"HEIDI – Una herramienta de apoyo para el diseño de redes de comunicación"
H. Cancela, M. Urquhart.
Technical report INCO 93-01, 1993.

[CU 94]
Métodos Monte Carlo en la estimación de la confiabilidad en redes de comunicaciones - Estado del arte-
H. Cancela, M. Urquhart
Reporte Técnico INCO 94.06, InCo, Facultad de Ingeniería - 1994

[CU 00]
"Adapting RVR simulation techniques for residual connectedness network reliability models"
H. Cancela y M. E. Urquhart.
XXVI Latin American Conference On Informatics (CLEI'2000), CLEI – ITESM, 2000.

[CUR 93]
Una Herramienta de apoyo para el diseño de Redes de Comunicaciones.
H.Cancela, M. Urquhart, G. Rubino.
InCo - Facultad de Ingeniería, URUGUAY, IRISA - INRIA, Rennes, FRANCIA

[CP 01]
Diameter constrained network reliability: exact evaluation by factorization and bounds.
H. Cancela y L. Petingi
Proceedings of ICIL'2001 (International Conference on Industrial Logistics), páginas 359-366, Okinawa, Japan, 9-12 de Julio de 2001.

[CP 03]
"Diameter constrained network reliability: exact evaluation by factorization bounds"
Héctor Cancela y Louis Petingi.
DCR-INCO 0103, 2003.

[CP 04]

Reliability of Communication networks with delay constraints: computational complexity and complete topologies.
H. Cancela y L. Petingi
International Journal of Mathematics and Mathematical Sciences, 2004(29-32): 1551-1562, 2004.

[EKR 92]
A Monte Carlo method based on antithetic variates for network reliability computations.
Mohamed El Khaidiri y Gerardo Rubino.
Publicación interna, IRISA, 1992.

[EKMR 91]
Parallel estimation of 2-terminal network reliability by a crude Monte Carlo technique.
M. El Khaidiri, R. Marie y G. Rubino.
Sixth Internacional Symposium on Computer and Information Sciences, Antalaya, Turquía - Ocubre de 1991.

[ER 92]
"A Monte Carlo method based on antithetic variates for network reliability computations"
Mohamed El Kahdiri y Gerardo Rubino
IRISA, Universidad de Rennes, Francia, 1993.

[Fish 86]
A comparison of four Monte Carlo methods for estimating probability of s-t conectedness.
G. S. Fisherman
IEEE Trans. On Reliability R-35(2), Diciembre de 1986

[Maut 00]
Proyecto de Taller V: Método RVR en la simulación de medidas de confiabilidad en redes.
Antonio Mauttone.
InCo, Febrero de 2000.

[PU 96]
Algorithms for the computation of communication network vulnerability indexes
L. Petingi, M. Urquhart.
Reporte Técnico InCo 96.04, InCo, Facultad de Ingeniería - 1996

:: Apéndice F

:: Optimización: CDA Optimizer

F.1. Introducción

En éste apéndice presentaremos en detalle algunos aspectos del diseño, desarrollo y posterior análisis de resultados referentes al problema de optimizar la confiabilidad diámetro acotada en una red.

Para comenzar haremos un desarrollo relativamente formal del problema que encararemos desde el punto de vista de la optimización combinatoria. Para ello plantearemos un modelo para las redes con que trabajaremos basado en las redes P2P, para lo que utilizaremos grafos completos. En base a este planteo, describiremos algunas particularidades de nuestro problema particular, las cuales serán consideradas para el desarrollo del algoritmo de optimización que diseñamos y posteriormente implementamos. Precisamente, esta primera sección se cierra con la presentación del algoritmo de optimización en alto nivel.

A continuación incluiremos todo el desarrollo del diseño realizado para la aplicación de optimización, desde los requerimientos planteados para la misma, pasando por los diagramas de clase más importantes, y finalizando con un listado de posibles extensiones y mejoras factibles de hacer sobre la aplicación más algunos bugs detectados en la misma.

La última gran sección de este apéndice es la referente a la presentación y análisis de los resultados obtenidos, para lo cual tomaremos algunos ejemplos particulares sobre los que analizaremos en detalle los resultados de la ejecución del algoritmo de optimización.

F.2. Algoritmo de optimización

En esta sección presentaremos todo el trabajo realizado previo a la implementación del algoritmo de optimización de la confiabilidad diámetro acotada, comenzando con una presentación propiamente del problema que se pretende resolver hasta llegar a un diseño de alto nivel (en forma de pseudocódigo) del algoritmo a implementar, pasando por un análisis teórico del problema y las particularidades del mismo que se usaron para derivar el algoritmo finalmente planteado.
El mismo no forma parte de la artillería "común" de algoritmos de optimización combinatoria (como podrían ser las redes neuronales, caminos de hormigas, entre otros) sino que es un algoritmo creado "a medida" de nuestro problema lo que justamente permite explotar al máximo algunas propiedades particulares del mismo. Esto, además de la simplicidad y el buen comportamiento en cuanto a tiempo de cálculo que demostró tener en algunas pruebas preliminares, fueron los factores claves para la elección del algoritmo.

F.2.1. Presentación del problema

El trabajo hasta el momento se ha centrado en la evaluación de diferentes medidas de interés sobre los grafos de forma abstracta, haciendo nada más que algunas menciones a como se podrían aplicar las ideas manejadas en problemas concretos, como es el caso de las redes P2P.
En estos sistemas el mecanismo más común para el descubrimiento de la información es la difusión implementada como una búsqueda BFS, mediante la cual un nodo manda un pedido de información a todos sus vecinos y cada uno de ellos, en caso de no poseer la información buscada, repite este procedimiento reenviando el pedido a todos sus vecinos.
Una medida interesante en este entorno es la cantidad máxima de nodos por la que puede pasar un paquete de pedido de información antes de ser descartado (o vista de otra forma, la cantidad de reenvíos que puede sufrir un paquete antes de su descarte) que se conoce como time-to-live o TTL. Aumentar el TTL implicaría aumentar el tráfico de paquetes de la red, disminuyendo entonces la eficiencia global de la misma, mientras que una disminución del mismo podría dejar como inaccesible informaciones que en realidad si se encuentran disponibles en el sistema.
Otro parámetro importante es la cantidad de conexiones promedio que realiza cada nodo de la red, que actúa de forma similar que el TTL: un aumento de la misma radica en un aumento de la carga de la red, mientras que una disminución podría mostrar como no accesible información a la que se podría acceder con la posibilidad de más conexiones.

En estos sistemas hay un claro tradeoff entre la minimización de la carga de la red y la probabilidad de que un nodo pueda acceder a información que brinda otro nodo. Este problema es el que pretendemos abordar desde la perspectiva de la optimización combinatoria utilizando algún método que resulte particularmente interesante y eficiente en esta clase de problemas particular. Para ello tomaremos como base los algoritmos de evaluación de las dos medidas de interés en este contexto: cantidad de paquetes enviados por los nodos de la red (o carga de la red) y la confiabilidad diámetro acotada de la red.
El problema se puede plantear de una manera más formal desde dos enfoques diferentes:

- Maximizar la probabilidad de que los nodos puedan comunicarse entre sí sin superar una restricción dada en la cantidad de paquetes enviados por los nodos de la red.
- Minimizar la carga en la red sin pasar una cota mínima dada en la calidad de servicio que se brinda.

El costo de una red, puede ser medido (desde el punto de vista del proveedor del servicio) a través de la cantidad de conexiones que se permiten a cada usuario o nodo. La cantidad de conexiones en este contexto puede ser vista como una medida representativa del ancho de banda al que tienen acceso los usuarios.

Por otro lado, la calidad del servicio puede ser pensada de más de una forma:

1) Por el flujo de información que se permite ingresar al usuario en la red, o sea, la cantidad de pedidos de información (que podríamos medir como la cantidad de paquetes inducidos por estos pedidos) generados por el usuario.
2) Por la proporción de información disponible en la red a la que es posible acceder desde los nodos, que puede medirse por la confiabilidad diámetro acotada de la red.

Como queda claro al repasar los enfoques planteados, tomaremos la segunda de estas opciones como medida de la calidad del servicio brindado al momento de presentar formalmente el problema de optimización que resolveremos.
El mismo podría ser planteado de la siguiente manera para cada unos de los enfoques mencionados:

$$\begin{cases} \text{Maximizar Confiabilidad Diametro Acotada} \\ \text{Carga} < L \end{cases} \quad \text{(E1)}$$

$$\begin{cases} \text{Minimizar Carga} \\ \text{Confiabilidad Diametro Acotada} > S \end{cases} \quad \text{(E2)}$$

Utilizaremos a lo largo de este documento las siguientes abreviaciones para referirnos a los parámetros involucrados en el problema:

- $CDA \in [0,1]$: Confiabilidad diámetro acotada
- $C \in \Re^+$: Carga de la red
- $CCP \in \Re^+$: Cantidad de conexiones promedio por nodo
- $D \in \aleph^+$: Diámetro máximo permitido para considerar un camino como operacional (o TTL).
- $P \in [0,100]$: Porcentaje de nodos comunicados por caminos de largo menor que un cierto *D*.

Los problemas anteriores pueden ser planteados en un formato más matemático de la siguiente forma:

$$\begin{cases} Max\ CDA \\ S.A.\ C < L \end{cases} \quad \text{(E1')}$$

$$\begin{cases} Min\ C \\ S.A.\ CDA > S \end{cases} \quad \text{(E2')}$$

En donde $L \in \aleph$ representa el límite impuesto para la carga en la red, mientras que $S \in [0,1]$ es la cota mínima en la calidad de servicio ofrecido.

En definitiva, lo que buscamos con este desarrollo es encontrar la pareja (*CCP*, *D*) que sea solución de uno de los problemas (E1') y (E2') planteados.
Más aun: podríamos plantear e intentar solucionar el problema multiobjetivo de maximizar la *CDA* y minimizar la *C*, enfoque que se dejará de lado en presente trabajo.

En este contexto podemos pensar nuestro problema de optimización de la siguiente forma:

- **Parámetros de entrada:**
  - $N \in \aleph^+$ → Cantidad de nodos de la red.
  - $L \in \aleph^+$ → Límite de carga de la red.

  - $CPP \in \aleph^+$ → Cantidad de pedidos promedio por nodo.
- **Variables de decisión:**
  - $CCP \in \aleph^+$ → Cantidad de conexiones promedio por nodo.
  - $D \in \aleph^+$ → Largo de los caminos operacionales válidos de la red.
- **Espacio de problemas:**
  - Podemos pensar nuestro espacio de problemas como una matriz *G* de *N-1* x *N* en donde la *CCP* varía en las ordenadas y *D* en las abscisas.
- **Salida:**
  - Pareja de valores (*CCP*, *D*) que maximizan la confiabilidad diámetro acotada -*CDA*- cumpliendo la restricción de carga *C* < *L*.

Cabe una aclaración con respecto al espacio de problemas que elegimos para nuestro problema. Se tomaron *CCP* y *D* variando en los Naturales, aunque esta es una elección arbitraria ya que se podrían haber tomado en el dominio de los Reales. La justificación de esta elección es la simplificación del problema de optimización y posibilitar la utilización de optimización combinatoria para la resolución del mismo.

F.2.2. Algunas particularidades de nuestro problema

Para empezar cabe notar que dado que la *CDA* de una red está acotada por el valor 1, podemos tener más de una pareja de valores en la matriz que cumplan las condiciones para ser solución del problema E1': todas aquellas parejas para las cuales la evaluación de la confiabilidad diámetro acotada sea 1.
Afinando un poco esta observación, todos los puntos de la matriz para los que obtengamos *CDA* = 1 como producto de la evaluación, en realidad son un valor muy cercano a 1 que nuestro algoritmo de evaluación (que no es exacto) redondea en 1, por lo que en caso de requerirse un afinamiento del conjunto de soluciones obtenido basta con aumentar la precisión de la evaluación.

Analizando más detenidamente las medidas en consideración en nuestro problema, podemos detectar algunas propiedades particulares del mismo:

- La función de confiabilidad es creciente para valores de *CCP* crecientes y un valor fijo de *D*.
- La función de confiabilidad es creciente para valores decrecientes de *D* y un valor fijo de *CCP*.

En base a estas consideraciones, es claro que para cada fila (valor fijo de $CCP=CCP_0$) de la matriz de problemas podemos encontrar un valor de $D=D_0$ que es límite en cuanto al cumplimiento de la restricción de carga, o sea:

$$\Rightarrow \phi_{c\arg a}(G[CCP_0, D_0]) < L$$

La existencia de este valor está dada por la monotonía de la función de carga ($\phi_{c\arg a}$):

$$D_i < D_{i+1} \Rightarrow \quad \phi_{c\arg a}(G[CCP_0, D_i]) < \phi_{c\arg a}(G[CCP_0, D_{i+1}])$$

Este valor de diámetro de camino o TTL -D_0- será también el óptimo para nuestro problema de optimización restringido a la fila definida por $CCP=CCP_0$, debido a que la propiedad de monotonía también se cumple para la ϕ_{DA}, por lo que cualquier valor de *D* menor que D_0 resultará en un valor de confiabilidad diámetro acotada menor y cualquier valor mayor violará la restricción de carga y quedará excluido del conjunto de soluciones.
De esta forma podemos definir una primer "frontera" de potenciales soluciones, formada por cada uno de los estos óptimos $D_{i,0}$ de los subproblemas definidos, para cada uno de los *i* valores de *CCP* (*i*= 1,…, *N*-1)

Un refinamiento de esta frontera puede hacerse considerando la monotonía de la función de carga respecto a *CCP*. O sea, para un valor fijo de $D=D_0$, se cumple que:

$$CCP_i < CCP_{i+1} \Rightarrow \quad \phi_{c\arg a}(G[CCP_i, D_0]) < \phi_{c\arg a}(G[CCP_{i+1}, D_0])$$

De forma que si para un conjunto de valores de *CCP* se cumple que el valor de *D* óptimo en cada una de las filas coincide, solo el mayor de estos valores - CCP_M - nos es de interés para nuestra solución ya que, según la monotonía de $\phi_{c\arg a}$ respecto a *CCP*, para cualquier valor mayor que CCP_M se violará la restricción de carga planteada para la optimización (si $\phi_{c\arg a}(G[CPP_{M+1}, D_0]) < L$ entonces el óptimo del subproblema en $G[CCP_{M+1}, D_0]$ coincidiría con el óptimo del subproblema en $G[CCP_M, D_0]$, lo que contradice que CCP_M sea el máximo valor de *CCP* para el valor particular de *D* en consideración), mientras que un valor menor que CCP_M no será optimo por la monotonía de la función de carga.

Entraremos más en detalle en estas particularidades de nuestro problema cuando desarrollemos el algoritmo de evaluación creado para el mismo, a partir de las siguientes secciones.

F.2.3. Propiedades teóricas de las funciones de evaluación

Nuestro algoritmo de optimización se basa en 4 propiedades básicas de las funciones de evaluación de la carga ($\phi_{c\arg a}$) y de la confiabilidad diámetro acotada (ϕ_{DA}) sobre el conjunto de grafos del problema.

F.2.3.1. Función de carga creciente para valores crecientes de *D*:

Cabe aclarar que la función de carga que utilizamos en los cálculos no es exacta y se basa en una fórmula analítica desarrollada pensando en un comportamiento “estándar” del mecanismo de distribución de los paquetes en el sistema (inundación basada en un algoritmo BFS). Será usada como una aproximación del comportamiento real del sistema y su calidad fue medida de forma empírica en las pruebas de validación de la herramienta de evaluación (ver **Apéndice E**). Como se podrá observar allí, la misma, para todos los casos evaluados, siempre retorna un valor dentro del intervalo de confianza definido para el Método de Monte Carlo.
Debido a todo lo mencionado y lo costosas que son, en materia de tiempo de ejecución, las evaluaciones reiteradas de la función de carga, en nuestro algoritmo de optimización utilizaremos este estimador de la carga en la red en lugar de utilizar los métodos de simulación.

$$D_i < D_j < N$$
$$\Rightarrow \phi_{c\arg a}(G[CCP_0, D_i]) \leq \phi_{c\arg a}(G[CCP_0, D_j])$$

Intuitivamente esta relación apoya la idea de que al aumentar el TTL de los paquetes (que en definitiva es lo que representa el aumento de *D*) aumentará la cantidad de paquetes enviados en la red. En otras palabras, aumentará la carga en la red.

F.2.3.2. Función de carga creciente para valores crecientes de *CCP*:

$$CCP_i < CCP_j < N-1$$
$$\Rightarrow \phi_{c\arg a}(G[CCP_i, D_0]) \leq \phi_{c\arg a}(G[CCP_j, D_0])$$

El aumento en la cantidad de conexiones permitidas (en promedio) por cada nodo, genera un aumento en la “la amplitud” de la búsqueda que realiza cada nodo en pos de la información de

interés. Teniendo en mente que en nuestro modelo el mecanismo de búsqueda es básicamente un broadcast (posiblemente con alguna mejora sobre el broadcast "puro"), este aumento en el grado de salida de cada nodo repercute en un aumento en la cantidad de paquetes que salen de los nodos. Este aumento será más notorio a medida que se vallan alcanzando más niveles de la búsqueda, siendo el crecimiento del mismo, exponencial en la profundidad de búsqueda (o sea en D), como lo confirma la fórmula para la función presentada arriba.

F.2.3.3. Función de confiabilidad DA creciente para valores crecientes de *D*:

$$D_i < D_j < N$$
$$\Rightarrow \phi_{DA}(G[CCP_0, D_i]) \leq \phi_{DA}(G[CCP_0, D_j])$$

Siguiendo el mismo razonamiento que el planteado para el crecimiento de la función de carga para valores crecientes de *D*, al aumentar el largo de los caminos "permitidos" o análogamente el tiempo de vida de los paquetes en la red, estamos aumentando la probabilidad de encontrar la información que buscamos, lo que muestra intuitivamente la idea del crecimiento de la confiabilidad diámetro acotada al aumentar el valor de *D*.
Para este resultado, al igual que para el que sigue, no vamos a presentar una formal. La misma es sencilla de realizar teniendo en cuenta que todos los estados conexos de la red con el largo de camino acotado por D, también lo son con largo de camino D+1.

F.2.3.4. Función de confiabilidad DA creciente para valores crecientes de *CCP*:

$$CCP_i < CCP_j < N-1$$
$$\Rightarrow \phi_{DA}(G[CCP_i, D_0]) \leq \phi_{DA}(G[CCP_j, D_0])$$

El aumento en la amplitud de la búsqueda que genera el aumento en la cantidad de conexiones que se permiten a cada nodo en promedio, planteado en el punto **2)**, también genera un aumento en la probabilidad de encontrar la información en la red. Esto surge en el aumento polinómico de los nodos que se visitan en una búsqueda de información, el cual mencionamos que se incrementaría de forma exponencial al ir aumentando los niveles de la búsqueda. Este aumento en la cantidad de nodos visitados es polinómico, ya que no estamos considerando la llegada más de una vez a un mismo nodo de un paquete con el mismo origen de búsqueda, aunque por el momento podemos despreciar este efecto.

F.2.4. Algoritmo de optimización

Entrando definitivamente en lo que es nuestro algoritmo, comenzaremos por plantear el problema a resolver, que de los dos planteados, será el siguiente:

$$\begin{cases} Max\ CDA \\ S.A.\ C < L \end{cases} \qquad \text{(E1')}$$

Este problema define donde el espacio de soluciones estará definido por las duplas (*CCP*, *D*) con *D* variando entre 1 y *N*-1 y *CCP* entre 1 y un valor $CCP_{max} < N-1$, definido por el usuario. La restricción de esta variable es para modelar el hecho de que no siempre es posible definir de forma arbitraria en una red, la cantidad de nodos a los que se conecta un nodo particular.

Para fijar la idea presentamos el espacio de soluciones de nuestro problema con su frontera para el caso *N*=20 y *L*=5.000. En gris claro se marcan los puntos que cumplen la restricción de

carga, en gris oscuro los que además pertenecen a la frontera y en negro el punto donde se da el óptimo.

CP \ D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	20	39	57	74	90	105	120	133	146	159	170	181	192	202	211	220	228	236	244
2	40	116	259	531	1.047	2.024	3.875	7.381	14.026	26.615	50.468	95.664	#####	#####	#####	#####	#####	#####	#####
3	60	231	715	2.093	6.007	17.134	48.756	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
4	80	383	1.532	5.885	22.382	84.897	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
5	100	574	2.817	13.446	63.791	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
6	120	802	4.679	26.718	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
7	140	1.068	7.225	48.055	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
8	160	1.373	10.563	80.217	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
9	180	1.715	14.800	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
10	200	2.095	20.045	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
11	220	2.513	26.404	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
12	240	2.968	33.986	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
13	260	3.462	42.899	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
14	280	3.994	53.249	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
15	300	4.563	65.145	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
16	320	5.171	78.694	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
17	340	5.816	94.005	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
18	360	6.499	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####

Fig. F.1: Espacio de Soluciones para N=20 y LC=5.000

La elección del problema (E1) y no del (E2) se debió básicamente a los costos de evaluar cada una de las funciones en consideración. Mientras que para la carga tenemos una fórmula analítica que comprobamos empíricamente su buen comportamiento, para la confiabilidad de la red tendremos que utilizar la simulación con el consecuente aumento de tiempo de cálculo que trae aparejado. La idea entonces es tratar de llevar un mínimo razonable el número de evaluaciones de la función de confiabilidad sobre los grafos del espacio de problemas.

Con esta idea en mente, el algoritmo pensado funcionaría básicamente de la siguiente forma:

1. Encontrar la frontera de cumplimiento de la restricción de carga ($C < L$)
2. Recorrer este conjunto candidato y obtener el punto del mismo que maximice *CDA*.

Para entrar en un análisis más detallado de este algoritmo, comenzaremos por definir más formalmente la idea de frontera a que nos estamos refiriendo.
Con frontera aludiremos al conjunto de puntos *F* del espacio de problemas tales que:

$$F = \{(CCP_F, D_F) \in G \mid \phi_{c\arg a}(G[CCP_F, D_F]) \le L \;\; \neg\exists D_i > D_F \mid \phi_{c\arg a}(G[CCP_F, D_i]) < L \;\; \neg\exists CCP_i > CCP_F \mid \phi_{c\arg a}(G[CCP_i, D_F]) < L \;\; \}$$

En el peor de los casos este conjunto estará integrado por un punto de cada línea (el mencionado límite de carga D_0 de cada una de las líneas), totalizando *N*, con *N* igual a la cantidad de nodos de la red. Este valor sería en definitiva la cantidad máxima de evaluaciones de la función de confiabilidad que se tendrían que hacer.

La forma que ideamos de encontrar esta frontera tratando de minimizar igualmente el número de evaluaciones de la función analítica de carga, es como sigue:

1.1. Recorremos consecutivamente cada una de las líneas del espacio de soluciones (*CCP* constante en cada una de ellas) comenzando con la correspondiente a *CCP* = *N* - 2.

1.2. Considerando que la fila que estamos recorriendo es la correspondiente a *CCP* = *i*, obtenemos el límite para *D* - D_i^0 - despejando el mismo de la fórmula de carga que se presentó en la sub-sección E.2.2.2, tomando como valor de la carga el valor *LC* indicado por el usuario como límite de carga permitido. Si $D_i^0 > D_{i+1}^0$ incluimos el punto (CCP_{i-1}, D_{i-1}^0) en *F*, y en caso contrario lo descartamos.

Este último paso de no incluir en la frontera de carga el punto límite de la fila CCP_i en caso de que éste se de para un valor menor o igual de *D* que el que se obtiene en la fila CCP_{i+1}, se justifica por el hecho de que ϕ_{DA} es creciente para valores crecientes de *CCP* (ver sub-sección F.2.3.3)

Un pseudocódigo para este diseño podría ser de la siguiente forma:

```
FUNCTION FronteraDeCarga(LC): F
    Dant := 0

    For CCP:=N-2 to 1
          A := LC/CPN*N
          Do:=(Log(1+((A*C*(N-2)-A*(N-1))/(C*(N-1)))))/
                Log(C*(1-(1/(N-1))))
          If Do > Dant
                Enqueue((CCP, Do), F)
          EndIf

          Dant := Do
    EndFor
ENDFUNCTION
```

Habiendo completado el conjunto de puntos del espacio de problemas que integran la frontera, queda por delante la tarea de buscar dentro de este conjunto el o los puntos que maximicen el valor de la función de confiabilidad diámetro acotada.
Para esto procederemos simplemente recorriendo este conjunto, evaluando la función en cada punto y comparando el valor obtenido con el mejor valor hasta el momento. En caso de que el nuevo valor sea mayor, pasará a ser el mejor valor y se almacena el punto que estamos evaluando. Al final de este proceso tendremos como resultado el punto óptimo para el problema.
Siguiendo la línea de antes, presentamos el pseudocódigo para esta parte del proceso de optimización:

```
FUNCTION MaximizarConfiabilidadDA(F):(Cmax, CCPr, Dr)
    Cmax := 0
    CCPr := 0
    Dr := 0

    While (Not IsEmpty(F))
          (CCP, D) := Dequeue(F)
          C := ΦDA(CCP,D)

          If C > Cmax
                Cmax := C
                CCPr := CCP
                Dr := D
          EndIf
    EndWhile
ENDFUNCTION
```

Para culminar con la presentación del algoritmo de optimización que utilizaremos, incluiremos el pseudocódigo correspondiente al algoritmo de optimización completo:

```
FUNCTION Optimizacion(LC):(Cmax, CCPr, Dr)
    F := FronteraDeCarga(LC)
    (Cmax, CCPr, Dr) := MaximizarConfiabilidad(F)
```

```
ENDFUNCTION
```

F.3. Objetivos y requerimientos de la herramienta de optimización

F.3.1. Objetivos

Planteado formalmente el problema a resolver, quedan implícitamente planteados algunos objetivos para la herramienta encargado de solucionarlo.

Para empezar, a diferencia de lo que es el algoritmo de evaluación que mediante la herramienta de Visualización permite su aplicación a grafos no completos (ver **Apéndice D**), para el caso del optimizador la entrada será restringida solamente a grafos completos, por lo que la entrada podrá ser diseñada como un conjunto de parámetros simples a partir de los cuales se podrá obtener la topología del grafo.

Vale considerar también el uso de los algoritmos de evaluación en el optimizador. Como mencionamos en el planteo del problema y en el desarrollo del algoritmo, las funciones de evaluación de mayor interés son las que calculan la confiabilidad diámetro acotada y la cantidad de paquetes enviados por los nodos en una instancia del grafo particular, aunque la aplicación permite también maximizar el porcentaje de nodos a distancia menor o igual que *D*.
Dado que es de interés que la aplicación pueda tomar como entrada grafos del mayor tamaño posible, en lugar de usar los algoritmos de simulación implementados para la evaluación de la carga se utilizará la fórmula para su estimación presentada en la sección anterior, de forma de reducir el tiempo de cálculo.

F.3.2. Requerimientos

Para dejar más en claro los objetivos que describimos informalmente en los objetivos, y agregar algunos otros puntos planteados por el usuario, a continuación presentamos una lista de requerimientos para la aplicación a desarrollar:

- Herramienta de optimización:
  Se busca una herramienta que resuelva el problema (E1') planteado anteriormente mediante optimización combinatoria, para una familia de grafos definida por los parámetros de entrada. La salida será entonces una o más instancias particulares de este grafo genérico, para las cuales se maximiza la *CDA* sin romper la restricción planteada para *C*.

- Maximizar el rango de problemas donde la aplicación del algoritmo de optimización es factible y eficiente de aplicar:
  Dado que es necesaria la aplicación de algoritmos de evaluación en un número importante de grafos, es importante que esta evaluación de las funciones de interés sobre cada grafo sea lo más eficiente posible.

- Retornar lo que definimos como frontera de carga como salida auxiliar de la ejecución del programa.

F.4. Diseño

F.4.1. Consideraciones generales de diseño

Esta aplicación, al igual que todas las restantes, está diseñada y desarrollada siguiendo los principios básicos de cualquier desarrollo de capas: encapsulamiento de funcionalidad mediante clases y separación de las mismas en capas independientes.

Las capas que definimos son las clásicas: Presentación, Lógica y Persistencia o Acceso a datos. Cabe aclarar que una parte importante de la lógica está formada por un paquete con la implementación de los algoritmos de evaluación, que no van a ser descritos en el presente documento más que cuando sea necesario para explicar la interacción del algoritmo de optimización con los mismos.

Dentro de lo que son los métodos o algoritmos implementados, se destaca nítidamente el algoritmo de optimización, el cual se basa en el pseudocódigo presentado más arriba. Dado la importancia del mismo, se presentará en una parte del desarrollo de la capa Lógica mediante un diagrama de colaboración.

En las siguientes subsecciones detallaremos consecutivamente el diseño de las capas de Presentación, Lógica y Persistencia, incluyendo como corolario en cada una de ellas el diagrama de clases de diseño correspondiente.

F.4.2. Capa de Presentación

La capa de Presentación en esta aplicación es de una gran simplicidad. Solo cuenta con el main de la aplicación, la ventana de la misma, y una interfaz para el acceso a algunas acciones sobre la ventana principal, de forma que esta implementa la mencionada interfaz.
La idea de ésta es dar acceso a la capa Lógica para poder actualizar las barras de progreso que se incluyeron en la ventana: una para el progreso global del algoritmo de optimización, y otra para el progreso de la evaluación de la instancia particular del grafo que se esté procesando en cada momento.

En definitiva, el diagrama de clases de diseño para la capa queda como se presenta a continuación:

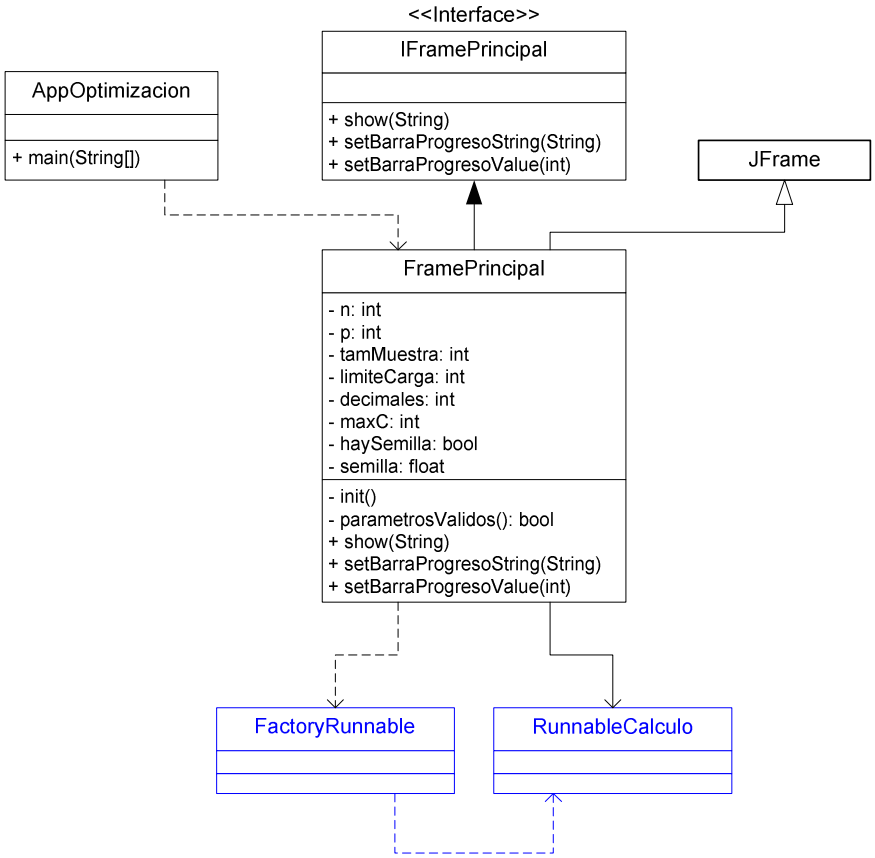


Fig. F.2: Diagrama de clases de la capa de presentación

Por más detalles sobre la capa de presentación y sobre como quedó construida la interfaz de ingreso de datos por parte del usuario, referirse al manual de usuario correspondiente a la herramienta de optimización en el **Apéndice G**.

F.4.3. Capa Lógica

Como se ve en la Figura F.2, el acceso a la lógica de optimización se realiza a través de dos clases: *FactoryRunnable* y *RunnableCalculo*. La primera, como su nombre lo indica, simplemente es una fábrica de instancias de la segunda, la cual se encarga de implementar el algoritmo de optimización descrito en las secciones anteriores.
Además de estas clases, solo se utiliza una clase auxiliar, el data type *Dupla*, el cual se utiliza para almacenar la información de cada uno de los puntos de nuestra espacio de problemas que son factibles de ser solución en los diferentes momentos del algoritmo: los puntos de la frontera de carga al definir la misma, y los puntos solución una vez recorrida la misma.

También podemos apreciar la interacción del optimizador con el evaluador, a través de las clases *Fi* y *GrafoEvaluación*. *Fi* es una clase abstracta y a nivel de instancias la interacción se da con la clase *FiDiametroAcotada*, que es la única función de evaluación que se utiliza de las implementadas mediante simulación.

Luego de estos breves comentarios sobre las clases que integran la capa Lógica, presentamos el diagrama de clases para la misma:

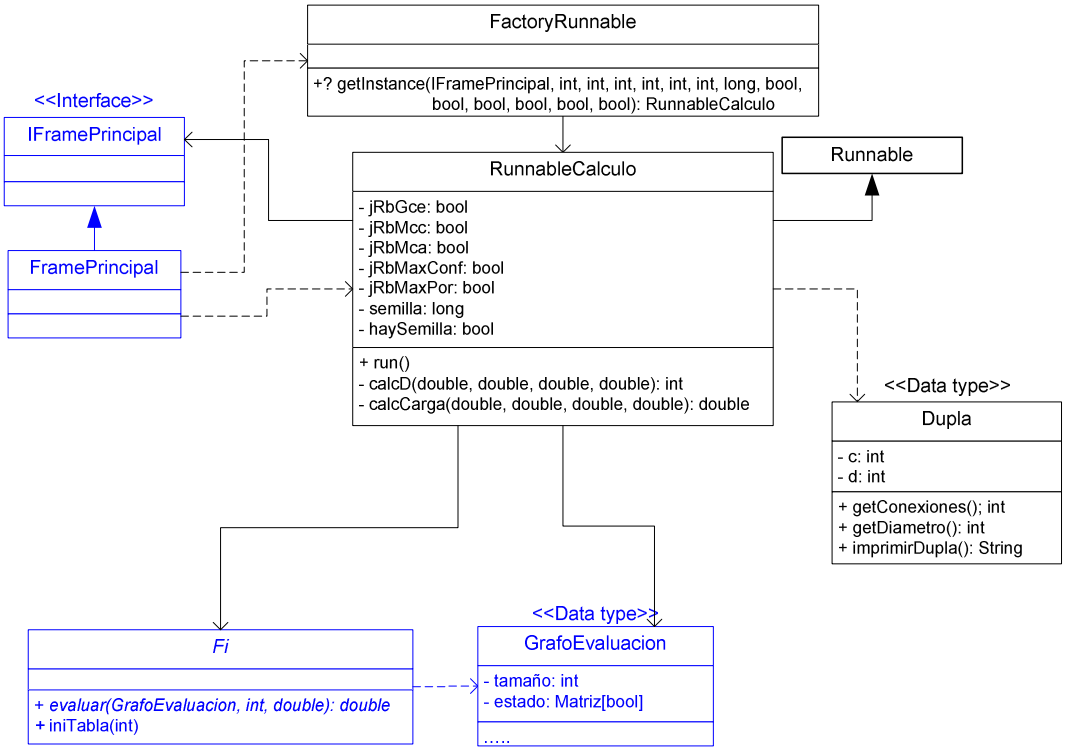


Fig. F.3: Diagrama de clases para la capa lógica

Entrando en lo que es la implementación en sí del algoritmo de optimización, el mismo sigue el pseudocódigo de la sección anterior, solo que se implementa todo el algoritmo de optimización en un solo procedimiento.

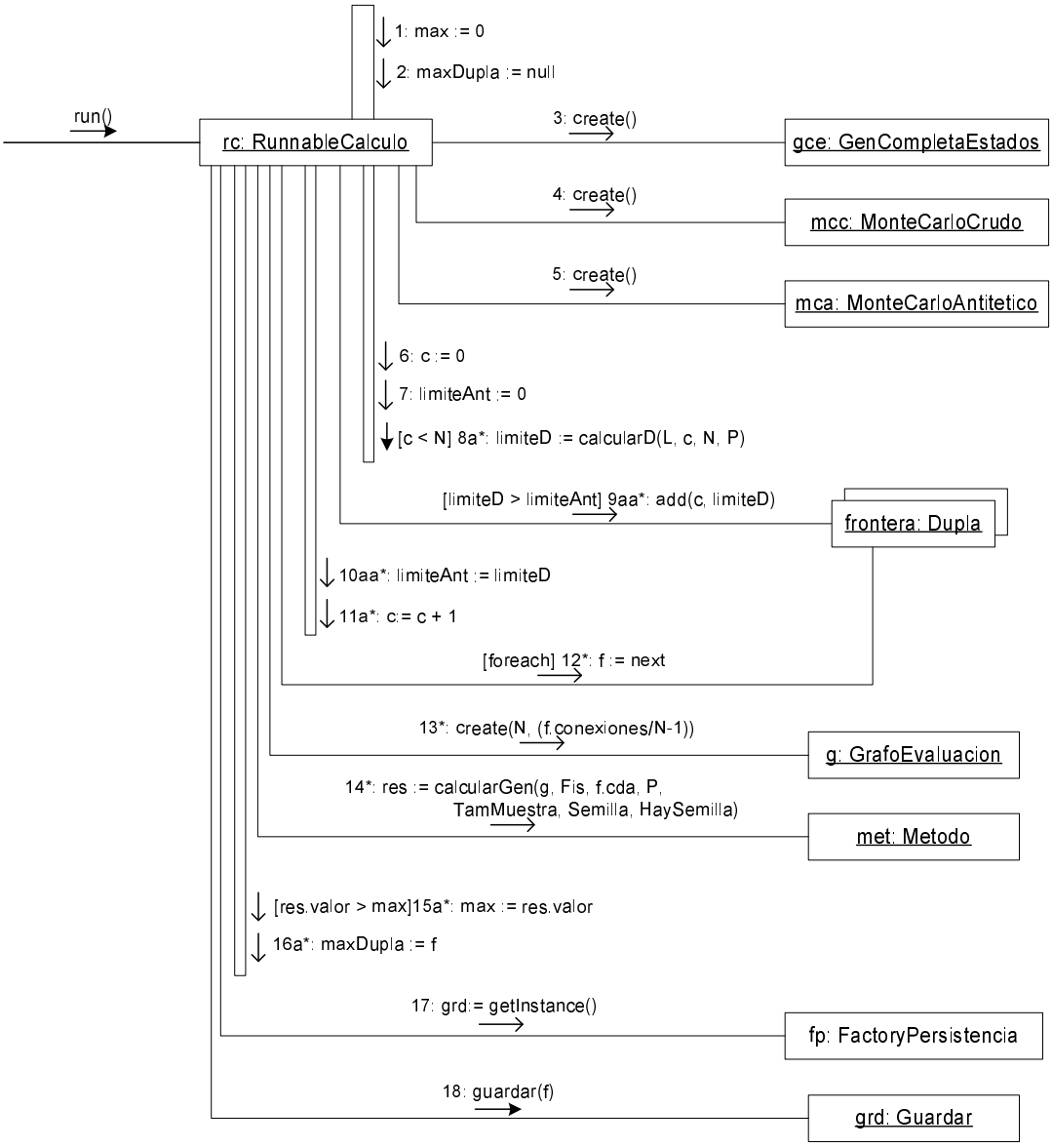


Fig. F.4: Diagrama de colaboración – RunnableCalculo:run()

Cabe hacer algunas aclaraciones referentes al diagrama de la figura anterior.
En primer término la rutina utiliza algunas variables globales a la clase (que hemos nombrado comenzando con mayúscula para diferenciarlas de las variables locales). Esto se debe a que esta clase se ejecuta como un hilo independiente en la aplicación, la misma solo puede recibir parámetros en el constructor, hecho que es una consecuencia de que la clase extienda a la clase *Runnable*.
Referente a la utilización de *Metodo*, repasando los diagrama de clases del evaluador vemos que la misma está definida como abstracta. En este diagrama hemos hecho un abuso de notación, utilizando la misma para hacer referencia a una de sus subclases, la que se determina al momento de la ejecución según las opción seleccionada por el usuario en la ventana de parámetros del programa.

F.4.4. Capa de Acceso a Datos

Para cerrar esta sección de diseño de la aplicación de optimización, presentaremos la estructura de la capa de acceso a datos.
Esta capa solo se encarga de crear un archivo con los datos obtenidos como resultado de la optimización, para lo cual se usan los servicios de las clases *PrintWriter* y *BufferedWriter*. El diagrama de clases para la capa quedaría entonces de la siguiente forma:

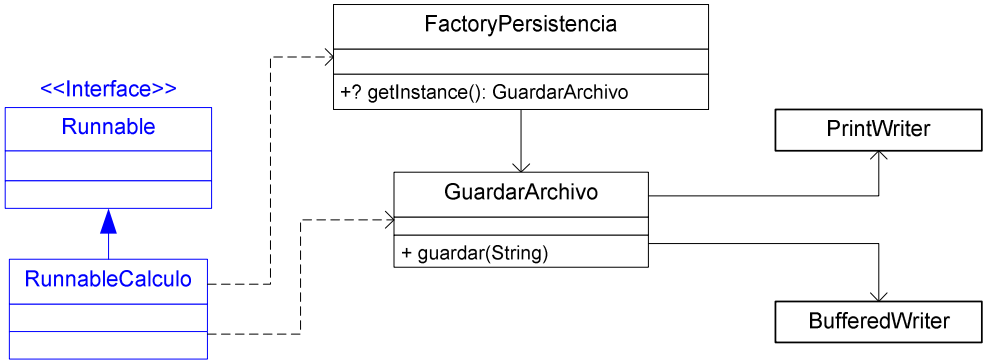


Fig. F.5: Diagrama de clases para la capa de acceso a datos

F.5. Estudio de fórmulas de carga

Como se ha venido mencionando, la utilización de los métodos de simulación estadística implementados en la herramienta de evaluación presentan la desventaja de los altos tiempos de ejecución que insumen. Si a esto sumamos la cantidad de evaluaciones de la función de carga que necesita nuestro algoritmo de optimización, que es del orden de o(N.Log(N)) para el caso en que se haga una búsqueda binaria en cada fila, es claro que necesitamos alguna forma más eficiente de evaluar la cantidad de paquetes enviados por los nodos de la red si queremos que nuestro algoritmo de optimización sea utilizable en la práctica.
En este sentido hemos optado por la búsqueda de funciones analíticas para la estimación de la función de carga, una de las cuales (aquella que en definitiva usamos en nuestro algoritmo) ya fue presentada en la sección E.2.2.2 del **Apéndice E**.
Sin embargo esta no fue la única función considerada. Concretamente se estudiaron en distintos momentos del proyecto, tres funciones de carga que presentamos a continuación:

Función 1:
Si asumimos que cada nodo *v* envía cada paquete que le llega por *CCP*-1 conexiones y que este proceso se repite hasta *D* veces, la cantidad de paquetes generados por un nodo estaría dada por:

$$\begin{aligned}\Rightarrow C_v &= CCP(1+(CCP-1)+(CCP-1)^2+\ldots+(CCP-1)^{D-1}) \\ &= \sum_{i=1}^{D}(CCP-1)^i = \sum_{i=0}^{D-1}(CCP-1)^{i+1} \\ &= \frac{(CCP-1)^{D+1}-1}{CCP-2}\end{aligned}$$

Por lo que la cantidad de paquetes generados por todos los nodos de la red estaría dada por:

$$\Rightarrow C_{F1} = N\frac{(CCP-1)^{D+1}-1}{CCP-2}$$

Función 2:
Como se mostrará más adelante, la fórmula 1 presenta un problema importante debido a que subestima el valor real (consideraremos como “valor real” el estimado por la simulación mediante el método de Monte Carlo), por lo que la utilización de esta función en el algoritmo de optimización podría provocar el retorno de un valor óptimo que en realidad no cumple con la restricción de carga impuesta.
En post de solucionar este inconveniente, una primera intención probada fue forzar a la función anterior a sobreestimar el valor calculado mediante el MCC, para lo que se intentó disminuir el valor del denominador en la fórmula 1, obteniéndose la siguiente:

$$\Rightarrow C_{F2} = N\frac{(CCP-1)^{D+1}-1}{CCP-1}$$

Función 3:
Si el número promedio de conexiones permitidas a cada nodo en promedio es *CCP*, y si todas las aristas del grafo son independientes, entonces la probabilidad de que una arista cualquiera esté activa independientemente del estado de las demás es *CCP*/*N*, con *N* siendo la cantidad de nodos de la red.
En concreto, cuando un paquete sale de un nodo v_i y llega a otro nodo v_k por la arista v_i-v_k , el nodo v_k reenviará el mensaje a todos sus adyacentes excepto a v_i. El nodo v_k tiene entonces *N*-2 (el propio nodo v_k tampoco cuenta) posibles vecinos distintos de v_i y cada uno de ellos existe

realmente con probabilidad *CCP*/*N*-1. Por lo tanto, el número medio de conexiones que esperamos desde v_k es *CCP*(*N*-1)/*N*.
En definitiva, un mensaje enviado desde un nodo *v*, va a generar C(1+ C(N-1)/N + (C(N-1)/N)^2 + ... + (C(N-1)/N)^(D-1)) mensajes, lo que en se reduce a la siguiente fórmula aproximada para la cantidad de paquetes enviados por los nodos de la red:

$$\Rightarrow C_{F3} = N.CPN\left(\frac{1-CCP^{D}(1-(1/N-1)^{D})}{(N-1)-(N-2)CCP}\right)$$

Planteadas las tres fórmulas que estuvieron en consideración, presentaremos algunos resultados obtenidos para las mismas y que permitieron la elección de una ellas para ser usada para mejorar la performance del algoritmo de optimización.
El estudio realizado se basa en la consideración de un caso base, y a partir del mismo ir variando uno de los parámetros manteniendo fijo el resto de forma de observar el comportamiento de la función de carga evaluada con las tres funciones anteriores y mediante el método Monte Carlo Crudo.
Los resultados obtenidos para la variación de *D*, *N* y *CCP* se presentan respectivamente en las tres gráficas siguientes:

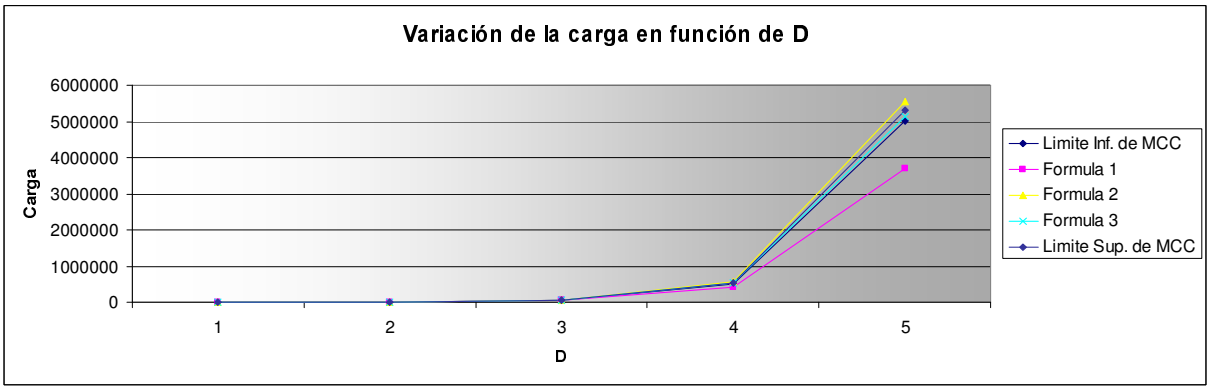


Fig. F.6: Variación de la carga al variar D

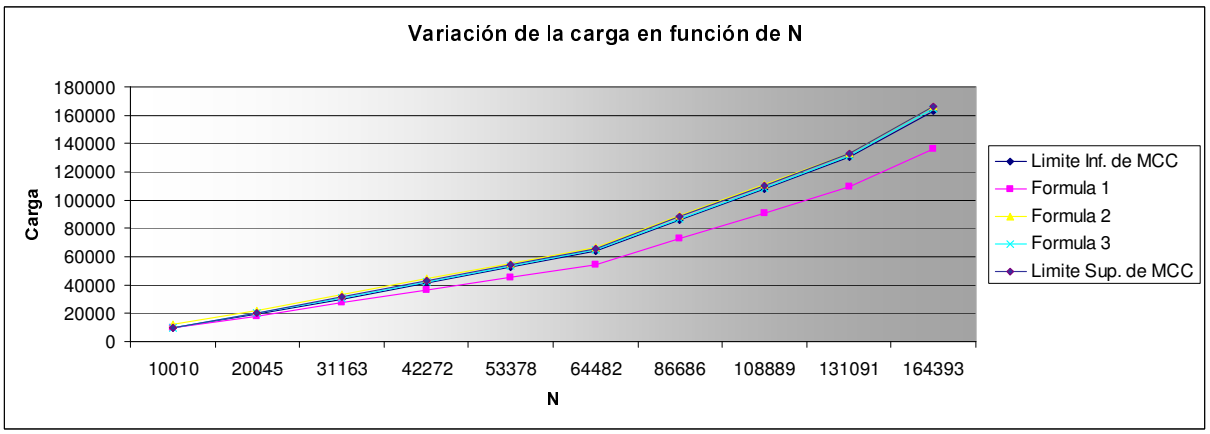


Fig. F.7: Variación de la carga al variar N

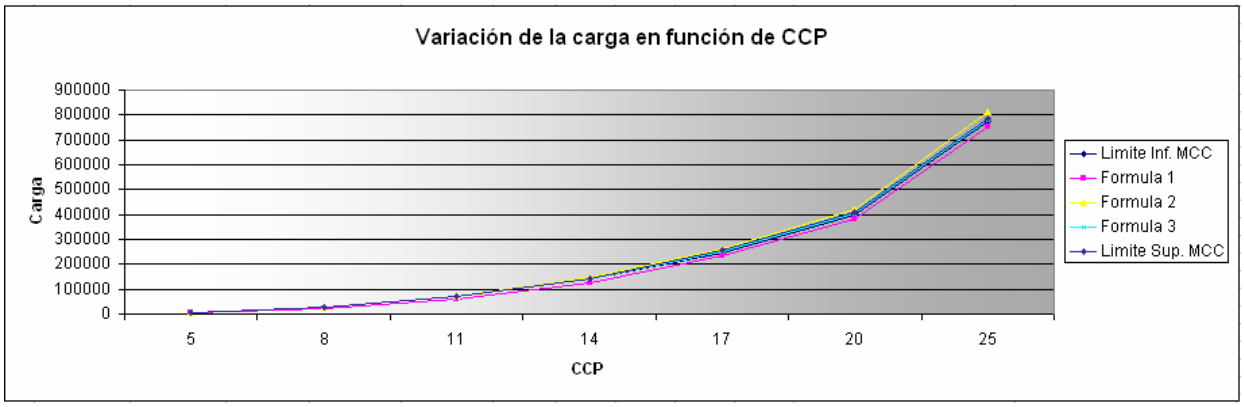


Fig. F.8: Variación de la carga al variar CCP

Los conjuntos de parámetros de entrada para las tres gráficas presentadas son:

- Variación de D (figuras F.6 y F.9): $N = 50$, $CCP = 10$, $CPN = 1$, $M = 1000$, $D = \{1, 2, 3, 4, 5\}$
- Variación de N (figuras F.7 y F.10): $D = 3$, $CCP = 10$, $CPN = 1$, $M = 1000$, $N = \{11, 20, 30, 40, 50, 60, 80, 100, 120, 150\}$
- Variación de CCP (figuras F.8 y F.11): $N = 50$, $D = 3$, $CPN = 1$, $M = 1000$, $CCP = \{5, 8, 11, 14, 17, 20, 25\}$

En los tres casos se observa que la función 1, como se mencionó arriba, subestima el valor retornado por el MCC; la función 2 sobre estima dicho valor; y la función 3 en algunos casos subestima y en otros sobre estima, pero siempre dentro del intervalo de confianza que hemos definido para el MCC.
Estos resultados primarios descartan la utilización de la función 1, mientras que entre las funciones 2 y 3 optamos por la última de ellas debido a la mejor calidad de sus resultados en el sentido de aproximación al valor retornado por el MCC.

Se realizó también un estudio más detallado de los resultados de las funciones comparando el porcentaje de error de cada una de ellas respecto al valor obtenido mediante el MCC. Al igual que antes, este estudio se hizo para variaciones de los tres parámetros de interés y sus resultados se presentan en las siguientes tres gráficas:

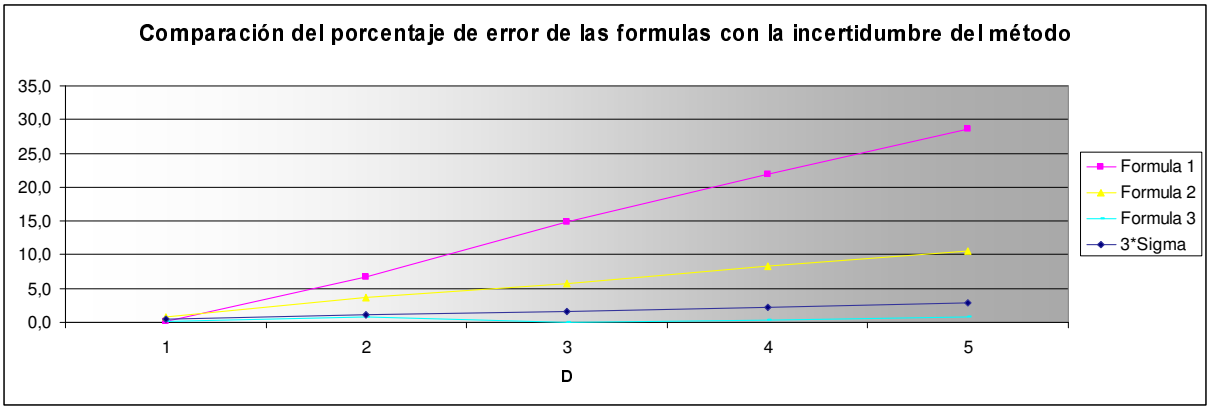


Fig. F.9: Estudio de errores en las funciones de carga al variar D

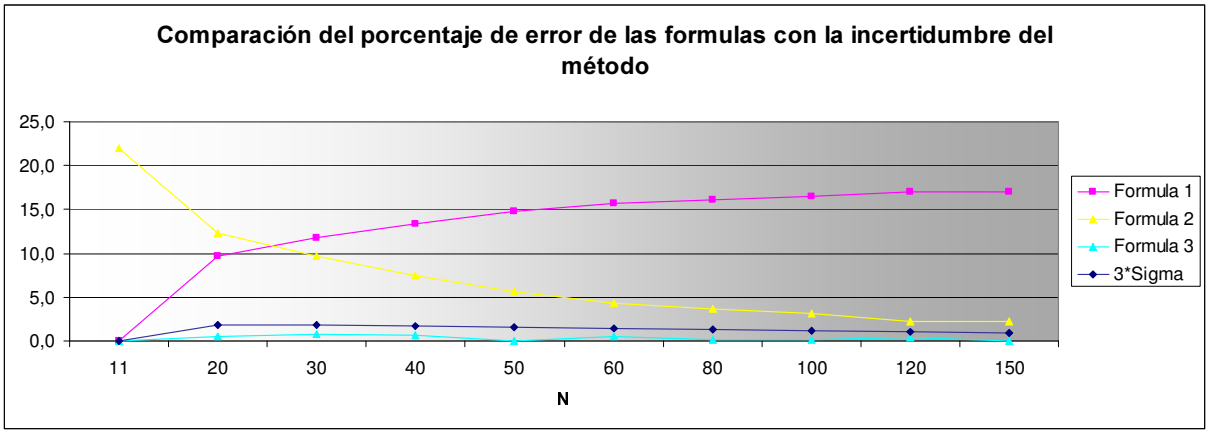


Fig. F.10: Estudio de errores en las funciones de carga al variar N

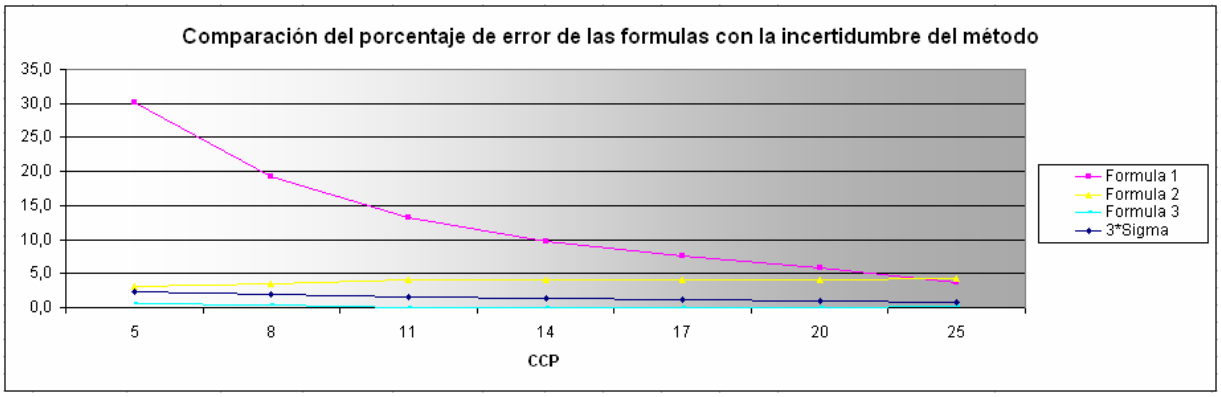


Fig. F.11: Estudio de errores en las funciones de carga al variar CCP

En todos los casos se observa que para la fórmula 1 el error va aumentando ante el aumento de los parámetros D y N, sin embargo va disminuyendo al aumentar CCP. Por otro lado para la fórmula 2 se observa un comportamiento similar, aunque menos pronunciado, aumentando el error al aumentar D o CCP, y disminuyendo al aumentar N. Por último para la fórmula 3 se comprueba la calidad de la misma observando que el valor absoluto de la diferencia entre el valor calculado por la misma y el retornado por el MCC, medido como porcentaje respecto al valor del MCC, nunca supera el 1%.

F.6. Resultados y conclusiones de la Optimización

Comenzaremos por explicar brevemente la forma en que se realizaron las pruebas cuyos resultados se analizarán seguidamente.
Los principales instrumentos que utilizaremos para el análisis de resultados son la matriz de carga del espacio de soluciones, y los resultados de la ejecución de la aplicación de optimización. Precisamente esta última, además de las combinaciones de valores para la cantidad de conexiones por nodo (CCP) y largo máximo de los caminos de la red (D) óptimos, retorna el conjunto de evaluaciones para la confiabilidad en todos los puntos que pertenecen a la frontera de carga del espacio de soluciones. Esta frontera estará definida por el parámetro de entrada que define justamente el límite de carga permisible en la red.
Por otra parte, la mencionada matriz de carga está definida por la cantidad de nodos de la red y la siguiente fórmula para la carga:

$$\Rightarrow C = N.CCP\left(\frac{1-CCP^{D}(1-(1/N-1)^{D})}{(N-1)-(N-2)CCP}\right)$$

la cual se comprobó empíricamente, es una muy buena estimación de la cantidad de paquetes enviados por los nodos de la red en un instante de tiempo.

El objetivo de las pruebas es estudiar el comportamiento de la frontera de carga y el óptimo retornado por la aplicación de Optimización, de forma de buscar de alguna forma validar los mismos y analizarlos en busca de algún comportamiento de interés.
Para validar la forma de la frontera, se compara la frontera retornada por la aplicación con una matriz de carga generada utilizando la fórmula de carga utilizada. También buscamos ver que para la pareja (CCP, D) retornada como optima, se cumple la restricción de carga planteada.

Debido a la naturaleza del problema y al método empleado para su resolución, parecería ser que la forma de la frontera de carga no varía de forma significativa ante variaciones proporcionales de N y el límite de carga de la red (LC).
Otro punto importante a tener en cuenta y relacionado con lo anterior, es que no sería necesaria la utilización de valores de N muy grandes para las pruebas, ya que según lo que mencionamos en el párrafo anterior, nos alcanzaría con fijar un valor de N y utilizar un conjunto variable de valores de LC para realizar un conjunto de pruebas útiles.
En este sentido los valores elegidos para la realización de estas pruebas primarias de los resultados de la optimización, fueron:

- N = 50
- LC = {20.000, 200.000, 2.000.000, 20.000.000}

Adicionalmente ser hizo una corrida particular para (N=500, LC=200.000) de forma de respaldar lo planteado arriba. Dado que en este caso todos los puntos en la frontera de carga tienen confiabilidad 0, se realizó otra ejecución para el caso (N=500, LC=20.000.000). El último caso que nos pareció interesante incluir es (N=100, LC=200.000) maximizando tanto la confiabilidad diámetro acotada con la función de Porcentaje.

Otro factor a tener en cuenta al momento de hacer las pruebas, es el referente a la limitación de los valores de CCP que podemos introducir en la red. No aplica en la realidad el hecho de que se pueda setear la cantidad de conexiones permitidas a cada nodo con un valor tan grande como sea necesario, estando el mismo acotado a un cierto valor umbral CCP_{max}.
La herramienta de optimización modela este hecho permitiendo en la entrada de la misma acotar este valor, por lo que incluiremos el mismo en las pruebas realizadas para estudiar un potencial cambio en el comportamiento de la solución entre los casos en que no se acote CCP y aquellos en que si se hace.

F.6.1 Casos de prueba

En esta sub-sección haremos una presentación de los resultados obtenidos en 7 casos de pruebas que parecieron de interés para el análisis de la herramienta.
Cada uno de ellos se presentará con el siguiente formato:

- Primero se incluye un snapshot de la ventana de retorno de la aplicación con los datos del punto óptimo encontrado.
- Luego incluimos la información completa de este punto óptimo.
- Finalmente se incluye el listado de puntos que integran la frontera de carga calculada durante la ejecución del algoritmo.

La información de estos dos últimos puntos es la que se incluye un archivo TXT que se genera como salida de la ejecución.

F.6.1.1. Caso 1: N = 50, LC = 20.000, sin restricción de CCP

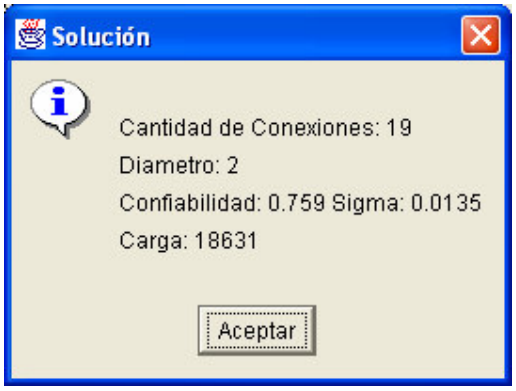


Fig. F.12: Resultado de la ejecución del optimizador para el Caso 1

```
N = 50
Limite de Carga= 20000
Pedidos = 1
Valor máximo de Conexiones = 48
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 1000
Semilla = 1
```

Cantidad de Conexiones: 48 Diametro: 1 Confiabilidad: 0.0 Sigma: 0.0 Carga: 2399	Cantidad de Conexiones: 3 Diametro: 5 Confiabilidad: 0.0010 Sigma: 0.0010 Carga: 16881
Cantidad de Conexiones: 19 Diametro: 2 Confiabilidad: 0.759 Sigma: 0.0135 Carga: 18631	Cantidad de Conexiones: 2 Diametro: 7 Confiabilidad: 0.0 Sigma: 0.0 Carga: 11446
Cantidad de Conexiones: 7 Diametro: 3 Confiabilidad: 0.027 Sigma: 0.00512 Carga: 19207	Cantidad de Conexiones: 1 Diametro: 49 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1557
Cantidad de Conexiones: 4 Diametro: 4 Confiabilidad: 0.0010 Sigma: 0.0010 Carga: 16086	

F.6.1.2. Caso 2: N = 50, LC = 200.000, sin restricción de CCP

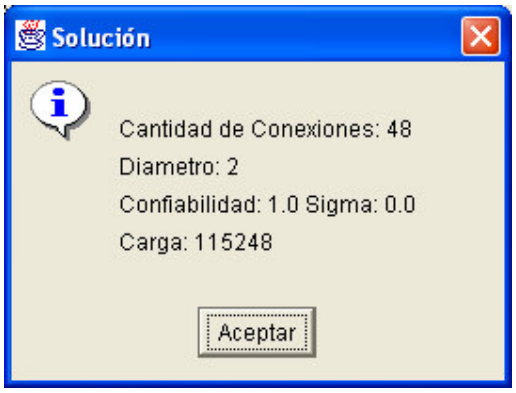


Fig. F.13: Resultado de la ejecución del optimizador para el Caso 2

```
N = 50
Limite de Carga= 200000
Pedidos = 1
Valor máximo de Conexiones = 48
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 1000
Semilla = 1
```

Cantidad de Conexiones: 48 Diametro: 2 Confiabilidad: 1.0 Sigma: 0.0 Carga: 115248	Cantidad de Conexiones: 3 Diametro: 7 Confiabilidad: 0.051 Sigma: 0.00696 Carga: 146385
Cantidad de Conexiones: 15 Diametro: 3 Confiabilidad: 1.0 Sigma: 0.0 Carga: 173702	Cantidad de Conexiones: 2 Diametro: 11 Confiabilidad: 0.0 Sigma: 0.0 Carga: 170082
Cantidad de Conexiones: 7 Diametro: 4 Confiabilidad: 0.903 Sigma: 0.00936 Carga: 132056	Cantidad de Conexiones: 1 Diametro: 49 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1557
Cantidad de Conexiones: 5 Diametro: 5 Confiabilidad: 0.633 Sigma: 0.0152 Carga: 180727	

F.6.1.3. Caso 3: N = 50, LC = 2.000.000, CCP_{max} = 5

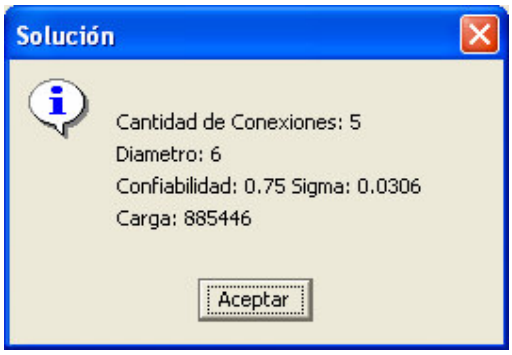


Fig. F.14: Resultado de la ejecución del optimizador para el Caso 3

```
N = 50
Limite de Carga= 2000000
Pedidos = 1
Valor máximo de Conexiones = 5
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 200
Semilla = 1
```

Cantidad de Conexiones: 5 Diametro: 6 Confiabilidad: 0.75 Sigma: 0.0306 Carga: 885446	Cantidad de Conexiones: 2 Diametro: 14 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1279720
Cantidad de Conexiones: 4 Diametro: 7 Confiabilidad: 0.465 Sigma: 0.0353 Carga: 971841	Cantidad de Conexiones: 1 Diametro: 49 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1557
Cantidad de Conexiones: 3 Diametro: 9 Confiabilidad: 0.09 Sigma: 0.0202 Carga: 1264837	

F.6.1.4. Caso 4: N = 50, LC = 20.000.000, CCP_{max} = 10

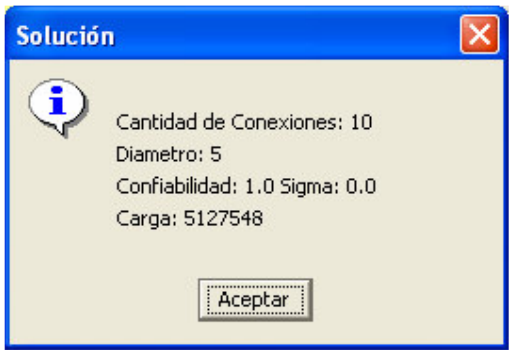


Fig. F.15: Resultado de la ejecución del optimizador para el Caso 4

```
N = 50
Limite de Carga= 20000000
Pedidos = 1
Valor máximo de Conexiones = 10
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 200
Semilla = 1
```

Cantidad de Conexiones: 10 Diametro: 5 Confiabilidad: 1.0 Sigma: 0.0 Carga: 5127548	Cantidad de Conexiones: 3 Diametro: 11 Confiabilidad: 0.095 Sigma: 0.0207 Carga: 1.09E7
Cantidad de Conexiones: 8 Diametro: 6 Confiabilidad: 0.985 Sigma: 0.00861 Carga: 1.35E7	Cantidad de Conexiones: 2 Diametro: 18 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1.88E7
Cantidad de Conexiones: 6 Diametro: 7 Confiabilidad: 0.93 Sigma: 0.0180 Carga: 1.49E7	Cantidad de Conexiones: 1 Diametro: 49 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1557
Cantidad de Conexiones: 4 Diametro: 9 Confiabilidad: 0.475 Sigma: 0.0353 Carga: 1.49E7	

F.6.1.5. Caso 5: N = 500, LC = 200.000, sin restricción de CCP

Fig. F.16: Resultado de la ejecución del optimizador para el Caso 5

```
N = 500
Limite de Carga= 200000
Pedidos = 1
Valor máximo de Conexiones = 498
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 200
Semilla = 1
```

Cantidad de Conexiones: 400 Diametro: 1 Confiabilidad: 0.0 Sigma: 0.0 Carga: 200000	Cantidad de Conexiones: 3 Diametro: 5 Confiabilidad: 0.0 Sigma: 0.0 Carga: 180222
Cantidad de Conexiones: 19 Diametro: 2 Confiabilidad: 0.0 Sigma: 0.0 Carga: 189638	Cantidad de Conexiones: 2 Diametro: 7 Confiabilidad: 0.0 Sigma: 0.0 Carga: 125719
Cantidad de Conexiones: 7 Diametro: 3 Confiabilidad: 0.0 Sigma: 0.0 Carga: 198764	Cantidad de Conexiones: 1 Diametro: 499 Confiabilidad: 0.0 Sigma: 0.0 Carga: 157806
Cantidad de Conexiones: 4 Diametro: 4 Confiabilidad: 0.0 Sigma: 0.0 Carga: 169087	

F.6.1.6. Caso 6: N = 500, LC = 20.000.000, sin restricción de CCP

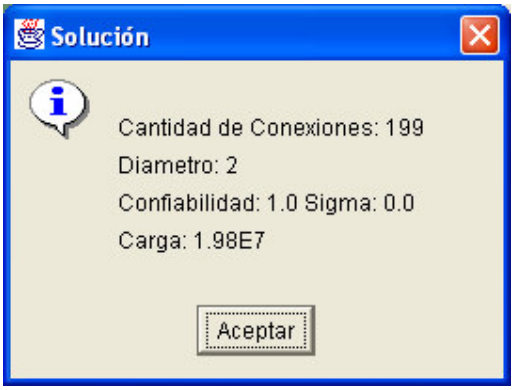


Fig. F.17: Resultado de la ejecución del optimizador para el Caso 6

```
N = 500
Limite de Carga= 20000000
Pedidos = 1
Valor máximo de Conexiones = 498
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 10
Semilla = 1
```

Cantidad de Conexiones: 498 Diametro: 1 Confiabilidad: 0.0 Sigma: 0.0 Carga: 249000	Cantidad de Conexiones: 5 Diametro: 6 Confiabilidad: 0.0 Sigma: 0.0 Carga: 9672393
Cantidad de Conexiones: 199 Diametro: 2 Confiabilidad: 1.0 Sigma: 0.0 Carga: 1.98E7	Cantidad de Conexiones: 4 Diametro: 7 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1.07E7
Cantidad de Conexiones: 33 Diametro: 3 Confiabilidad: 1.0 Sigma: 0.0 Carga: 1.84E7	Cantidad de Conexiones: 3 Diametro: 9 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1.45E7
Cantidad de Conexiones: 13 Diametro: 4 Confiabilidad: 1.0 Sigma: 0.0 Carga: 1.53E7	Cantidad de Conexiones: 2 Diametro: 14 Confiabilidad: 0.0 Sigma: 0.0 Carga: 1.59E7
Cantidad de Conexiones: 8 Diametro: 5 Confiabilidad: 0.1 Sigma: 0.1 Carga: 1.85E7	Cantidad de Conexiones: 1 Diametro: 499 Confiabilidad: 0.0 Sigma: 0.0 Carga: 157806

F.6.1.7. Caso 7A: N = 100, LC = 200.000, CCP_{max} = 20, para CDA

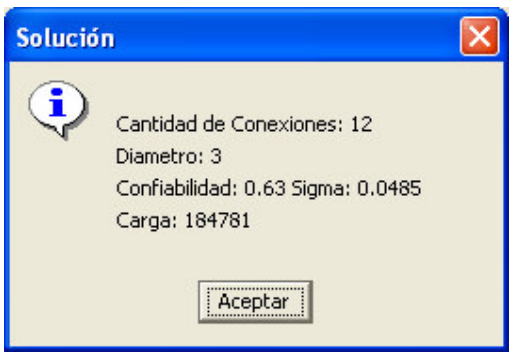


Fig. F.18: Resultado de la ejecución del optimizador para el Caso 7A

```
N = 100
Limite de Carga= 200000
Pedidos = 1
Valor máximo de Conexiones = 20
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 100
Semilla = 1
```

Cantidad de Conexiones: 20 Diametro: 2 Confiabilidad: 0.0 Sigma: 0.0 Carga: 41595	Cantidad de Conexiones: 4 Diametro: 5 Confiabilidad: 0.0 Sigma: 0.0 Carga: 131412
Cantidad de Conexiones: 12 Diametro: 3 Confiabilidad: 0.63 Sigma: 0.0485 Carga: 184781	Cantidad de Conexiones: 3 Diametro: 6 Confiabilidad: 0.0 Sigma: 0.0 Carga: 104318
Cantidad de Conexiones: 6 Diametro: 4 Confiabilidad: 0.0 Sigma: 0.0 Carga: 151041	Cantidad de Conexiones: 2 Diametro: 10 Confiabilidad: 0.0 Sigma: 0.0 Carga: 188639

F.6.1.8. Caso 7B: N = 100, LC = 200.000, CCP_{max}=20, para Porcentaje

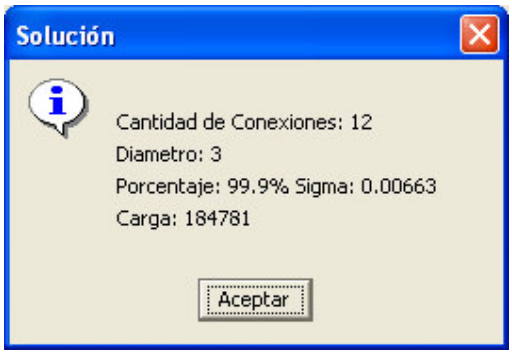


Fig. F.19: Resultado de la ejecución del optimizador para el Caso 7B

```
N = 100
Limite de Carga= 200000
Pedidos = 1
Valor máximo de Conexiones = 20
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 100
Semilla = 1
```

Cantidad de Conexiones: 20 Diametro: 2 Porcentaje: 98.6% Sigma: 0.0407 Carga: 41595	Cantidad de Conexiones: 3 Diametro: 6 Porcentaje: 83.6% Sigma: 0.679 Carga: 104318
Cantidad de Conexiones: 12 Diametro: 3 Porcentaje: 99.9% Sigma: 0.00663 Carga: 184781	Cantidad de Conexiones: 2 Diametro: 10 Porcentaje: 60.3% Sigma: 1.17 Carga: 188639
Cantidad de Conexiones: 6 Diametro: 4 Porcentaje: 98.8% Sigma: 0.101	Cantidad de Conexiones: 1 Diametro: 99 Porcentaje: 6.20% Sigma: 0.418

Carga: 151041	Carga: 6276
Cantidad de Conexiones: 4 Diametro: 5 Porcentaje: 93.4% Sigma: 0.385 Carga: 131412	

F.6.1.8. Conclusiones particulares de las pruebas

En las pruebas realizadas se pueden distinguir dos escenarios: aquellos en los que restringimos el *CCP* máximo y en los que no, surgiendo en cada uno de ellos las siguientes observaciones:

- **a)** En los casos que *CCP* tiene una cota máxima -CCP_{max}- , el óptimo se da la intersección de la frontera con *CCP*= CCP_{max} o en el escalón inmediatamente superior.
- **b)** En los casos que *CCP* no esta acotado, si la frontera corta a *C*=*N*-2 en un D_0>=2, entonces el optimo esta en (*N*-2, D_0); y si por el contrario la frontera corta a *C*=*N*-2 en *D*=1 o no la corta, el optimo estará en (CCL_0,2), siendo CCL_0 el determinado por el corte de la frontera con *D*=2

Notar que en esta sección llamamos frontera a todos puntos (C_i, D_0) tales que para todo D_1 > D_0 la carga esta por encima de la restricción de carga *LC*, con *i* variando de 1 a hasta *N*-2. Anteriormente habíamos definido frontera de otra forma, refiriéndonos en esas oportunidades a los puntos vértices de ésta frontera.

Entrando concretamente a las pruebas presentadas, los casos 1, 2, y 6 no tienen una restricción de CCP_{max}, y por tanto presentan lo observado en el punto **b)**. El caso 1 no tiene un corte de la frontera con *C*=*N*-2 en un *D*>=2 y por ende el máximo se da en el corte de ésta con *D*=2. En los casos 2 y 6, sí sucede que la frontera corta a *C*=*N*-2 en un *D*>=2 y por tanto el optimo se da allí.
Los casos 3, 4 y 7 tienen restricciones para CCP_{max} y por tanto aplican la observación **a)**. En los caso 3 y 4 particularmente el óptimo se da en el corte de CCP_{max} con la frontera, mientras que en el caso 7 en el escalón inmediato superior.

El caso 5 merece dos comentarios particulares: primero muestra la observación de que al variar proporcionalmente *N* y *LC*, la frontera se mantiene con la misma forma. En efecto la frontera es igual a la del caso 1, en la zona con *CCL* menor que 50. El segundo comentario es que este caso tiene la particularidad de que para la restricción de carga que se pasa como parámetro, ningún par (*CCL*,*D*) tiene una confiabilidad mayor a 0. Por lo que la aplicación devuelve un mensaje explicando que no hay un óptimo posible par ese caso.

El caso 6 tiene el mismo *N* que el anterior (500), pero se aumento *LC* 100 veces, para que hubiera duplas candidatas tuvieran CDA mayores a 0 (aumentando 10 veces seguía pasando el mismo fenómeno). No se limitó la carga y el optimo dio para *CCL*=199, ilustrando la observación **b)** dado que como el corte de la frontera en *N*-2 lo hace con *D*=1 el óptimo sucede en (199,2).
También se observa que en los demás puntos de la frontera la CDA es 0 o 1, ilustrando la observación que se hizo en el análisis de la evolución de que la CDA al variar *CCL* y *D* cambia muy rápidamente de 0 a 1.

El caso 7 además de ilustrar la observación **a)**, es interesante porque el resto de los vértices de la frontera tienen valores de CDA de 0 o muy cercanos a este. Esto es porque al limitar el CCP_{max} a 20, la CDA en (20,2) va a ser 0 o muy pequeña porque solo hay un 20% de las aritas (r=C/*N*-1) y por más que *D* sea 2, en casi todos los casos que se generan no hay diámetro confiabilidad con *D*=2. En el escalón “inmediato superior” de la frontera los grafos tendrán menos aristas pero al ser *D*=3, se obtiene una mayor CDA, luego para el resto de los casos, C es tan bajo que por mas de los valores de *D* aumenten, las CDA también están todas cercanas a 0.

Este caso motivo intentar crear un nuevo caso, el cual contradijera la observación b), pero no se encontró ningún valor de *CCL* en *D*=2, que hiciera que CDA sea los suficientemente próximo a 0, para que tomando como *LC* la carga de ese punto, sucediera que el optimo estuviera en *D*=3, lo que hubiere sido una excepción de b). En todos los intentos sucedió esta observación o por el contrario ningún candidato tenía CDA mayor a 0.

F.6.2. Conclusiones generales de la optimización

Según los resultados obtenidos las principales conclusiones que podemos sacar con respecto al comportamiento de nuestro algoritmo son las siguientes:

- Existe un óptimo trivial al problema que se encuentra en la posición (*CCP*=*N*-1, *D*=1) de nuestra matriz de soluciones. En la práctica este valor corresponde a una red en la que todos los nodos tienen un enlace de confiabilidad 1 (confiabilidad = *CCP*/*N*-1) con el resto de los nodos de la red, cosa bastante poco probable en una red mediana a grande. Debido a esto es que este valor trivial (no es necesaria la ejecución del algoritmo para saber que un óptimo se encuentra en ese punto, en los casos en que la confiabilidad máxima alcance a 1) no será tenido en cuenta para el análisis siguiente de los resultados.

- Independientemente de lo mencionado en el punto anterior, es notorio a partir de los resultados de las pruebas que el los valores óptimos tienden a estar en la zona de combinaciones de valores altos de CCP y valores bajos de *D*.
  El hecho de que el óptimo se encuentre en la zona indicada se justifica por el hecho de que para valores grandes de CCP los estados que se obtienen son en su mayoría completos o casi completos por lo que alcanza con valores chicos de *D* para que haya conectividad.
  Por otro lado, para la zona en que CCP toma valores chicos, por más que *D* tome valores elevados, los estados que se generen van a tener pocas aristas lo que aumenta la probabilidad de que el estado que se genere no sea conexo y para los que la confiabilidad es 0.
  Esta comprobación empírica ameritaría un estudio analítico en busca de las causas de este comportamiento de las soluciones, lo cual se escapa al alcance de los objetivos de este proyecto, quedando como una puerta abierta para la realización de trabajos futuros.

- Como se comentó en el punto **b)** de la sección anterior, en los casos en que se fija un valor para el máximo *CCP* posible -CCP_{max}-, se observa por lo general que el óptimo calculado se encuentra en (CCP_{max}, D_0), en donde D_0 está dado por el mayor valor de *D* para el que se cumple la restricción de carga, o en el "escalón inmediato superior" de la frontera.

- Asimismo para el caso en que *CCP* no sea acotado por el usuario habría según lo observado solo dos lugares posibles para el óptimo, como lo notábamos en la observación **a)** de la sección anterior.

- Para la frontera de carga de nuestra matriz de cargas, se comprueba el comportamiento esperado con una velocidad de crecimiento lineal en *D* y logarítmica en *CCP*, resultados estos que se conocían de antemano a partir del análisis de la fórmula usada para la construcción de la matriz.

- Análogamente a la matriz de carga construida, se podría construir una matriz de confiabilidades. Esta construcción no se ha realizado en esta primera instancia, ya que su construcción es muy costosa en tiempo de ejecución.

Cabe la aclaración que debido a la utilización de la fórmula aproximada para la evaluación de la función de carga y que la misma podría llegar a subestimar el valor exacto de la misma, el óptimo que se retorna producto del algoritmo podría llegar a no ser factible en el sentido del cumplimiento de la restricción de carga. Debido a esto se recomienda la evaluación de la

función de carga para los valores de *D* y *CCP* retornados como óptimos usando el CDA Evaluator de forma de comprobar su factibilidad como solución.
Este caso es muy poco probable, y de hecho en ninguna de las pruebas realizadas (tanto las documentadas arriba, como las pruebas unitarias que se hicieron durante el desarrollo de la aplicación) la aplicación retornó un óptimo no factible.

F.7. Trabajo futuro

F.7.1. Extensiones

Algunas extensiones al trabajo presentado en este documento fueron planteadas a lo largo del mismo. Éstas serán agrupadas en la presente sección, junto con algunas otras que no se han incluido en las secciones previas.

Para comenzar, aparece como interesante la realización de un estudio detallado de las causas de la ubicación del óptimo en la zona de valores de *C* grandes y *D* pequeños, ya que este comportamiento no se justifica de forma sencilla intuitivamente.

En lo que refiere a la aplicación de optimización en sí, se podrían pensar una serie de extensiones que incluyen:

- El armado de una salida gráfica que presente el espacio de problemas en donde se realiza la búsqueda y los resultados de las evaluaciones en los diferentes puntos en los que se realizan.

- Dar la posibilidad al usuario de elegir el nombre y el lugar donde guardará la información con los puntos de la frontera de carga encontrada.

F.7.2. Bugs

El único error detectado y no corregido en la aplicación es el referente a la posibilidad de que el resultado retornado no sea una solución factible del problema debido a la utilización de la fórmula analítica para la carga. El mismo requiere un estudio más detallado de la mencionada función de forma de ver realmente la posibilidad de que esto llegue a suceder, y en caso de que sí sea posible, encarar las medidas correctivas pertinentes que pueden pasar por la aplicación de un método para la evaluación de la función de carga en el punto antes de retornar el mismo como salida de la ejecución.

F.8. Referencias

[CE 98]
"Series-parallel reductions in Monte Carlo network reliability evaluation"
H. Cancela y M. El Khadiri.
IEEE Transactions on Reliability, 47(2):159-164, 1998.

:: Apéndice G

:: Manuales de Usuario

G.1. Manual de Usuario de GXL Translator

G.1.1. Instalación y Ejecución

G.1.1.1. Requerimientos

Se requiere J2SE Java Runtime Environment (JRE) 1.4.0 o superior.

G.1.1.2. Instalación

Para realizar la instalación de GXL Translator debe ejecutar el archivo llamado “Install.jar“. De esta manera se desplegará un asistente de instalación que lo asistirá en este proceso.

Dentro del proceso de instalación se pueden destacar dos etapas:

Una etapa en la que se pide al usuario que seleccione la ruta de instalación (Figura G.1) pudiendo ingresar la misma posicionado el cursor en el cuadro de texto o presionando el botón de “Escoger” para seleccionarla.

Fig. G.1: Selección ruta de instalación

Otra etapa, es la que pide que se seleccione los paquetes que desea instalar como se muestra en la Figura G.2. Dentro de los paquetes disponibles se encuentra

- Aplicación – paquete que contiene el ejecutable de la aplicación (la inclusión de este paquete es obligatoria)
- Documentación – manual de usuario y javadocs de la aplicación (la inclusión de este paquete es opcional)
- Fuentes – fuentes de la aplicación (la inclusión de este paquete es opcional)

Para incluir/excluir de la instalación a los paquetes opcionales se debe tildar/destildar el recuadro que aparece a la izquierda de cada uno de los paquetes.

Fig. G.2: Selección de paquetes a instalar

G.1.2.3. Ejecución

Una vez instalada la aplicación la misma tendrá la estructura de carpetas que se muestra a continuación:

- Bin – contiene el archivo de extensión JAR que es el ejecutable de la aplicación
- Doc – contiene el manual de usuario y los javadocs de la aplicación. En caso de no instalarse el paquete de “Documentos” esta carpeta no aparecerá.
- Src – contiene los fuentes de la aplicación. En caso de no instalarse el paquete de “Fuentes” esta carpeta no aparecerá
- Uninstaller – contiene un desinstalador de la aplicación

En la Figura G.3 muestra a modo de ejemplo como queda la estructura de la instalación si se selecciono como ruta de destino “D:\Archivos de Programa”

Fig. G.3: Estructura la instalación

Para ejecutar la herramienta debe posicionarse en la carpeta de nombre “bin” que se encuentra donde realizo la instalación y ejecutar el archivo llamado “GXL Translator.jar”

G.1.2. Funcionalidades

En esta sección describiremos como realizar conversiones de formato Heidi a GXL, y en el sentido inverso es decir de GXL a Heidi.

G.1.2.1. Traductor Heidi a GXL

Para generar un archivo GXL a partir de archivo Heidi, se debe seleccionar la opción “Traducir de formato Heidi a formato GXL”, como se muestra en la Figura G.4

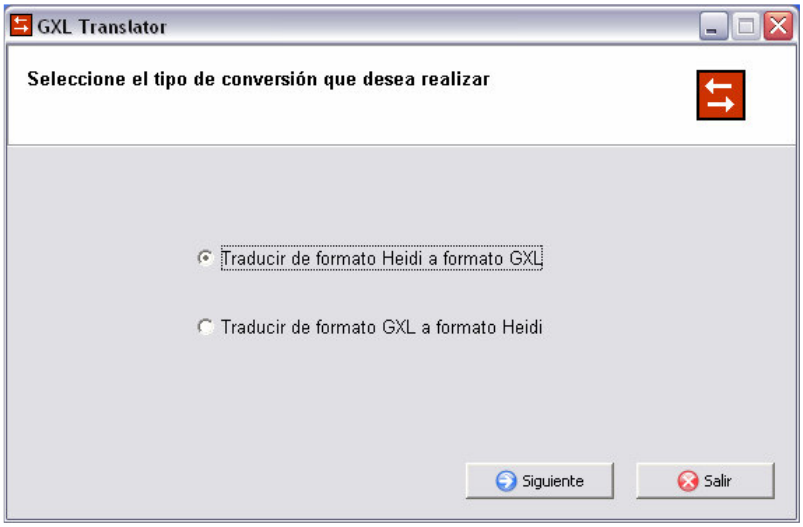


Fig. G.4: Selección de Conversión Heidi a GXL

En la siguiente ventana se deben seguir los siguientes pasos:

1- Ingresar el nombre absoluto, o relativo a la carpeta donde se encuentra la aplicación, del archivo Heidi a convertir, como se ve en la Figura G.5. Para facilitar esta tarea se puede oprimir examinar y seleccionar el archivo Heidi (ver Figura G.6).

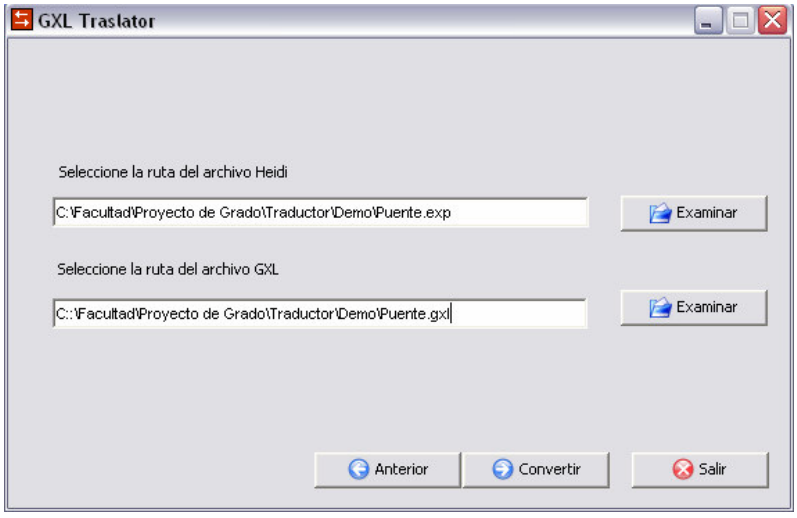


Fig. G.5: Ventana de conversión Heidi a GXL

2- Ingresar el nombre absoluto, o relativo a la carpeta donde se encuentra la aplicación, del archivo GXL que se va a crear. Si el archivo ya existe también se puede oprimir el botón examinar correspondiente, para seleccionarlo. En caso que el archivo no tenga extensión .gxl, la aplicación se la agregará automáticamente.

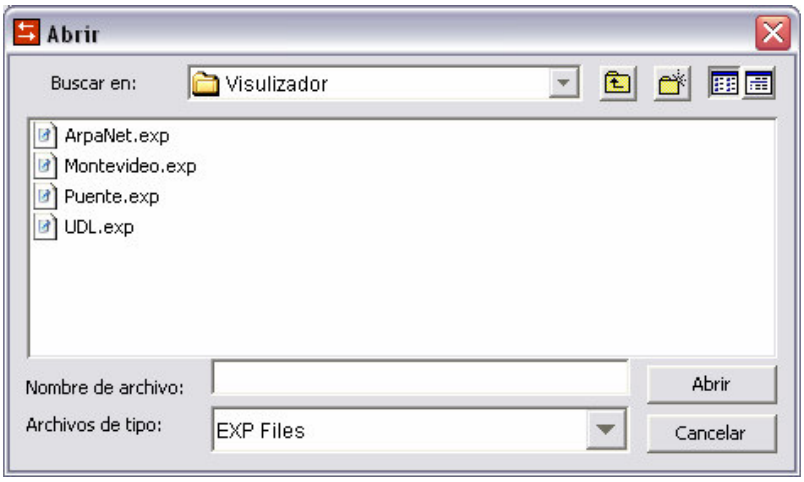


Fig. G.6: Selección del archivo Heidi

3- Oprimir el botón “Convertir”.
Si la conversión fue exitosa entonces se desplegará un mensaje notificándolo. Ver Figura G.7.

Fig. G.7: Confirmación de conversión exitosa de archivo GXL

Si hubo algún error en la conversión se desplegará el mensaje de error correspondiente. En la figura del ejemplo faltaba el tag FINNODO para al menos un nodo (ver Figura G.8)

En cualquiera de los dos casos, al clickear en botón Aceptar, vuelve a la ventana de traducción Heidi a GXL, para la siguiente conversión.

Fig G.8: La conversión falló por error de sintaxis en archivo Heidi

En cualquier momento de la conversión se puede oprimir el botón “Anterior”, para retornar a la ventana de selección de tipo de Conversión.

G.1.2.2. Traductor GXL a Heidi

Para generar un archivo Heidi a partir de archivo GXL, se debe seleccionar la opción “Traducir de formato Heidi a formato GXL”, como se muestra en la Figura G.9

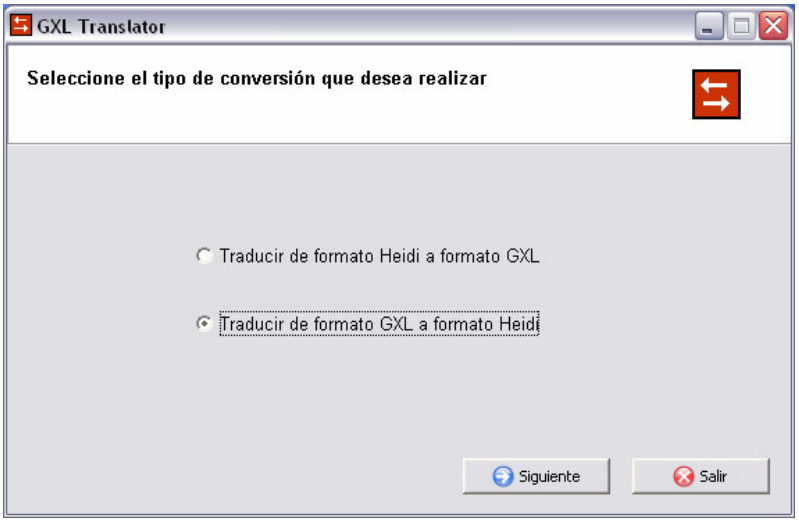


Fig. G.9: Selección de Conversión GXL a Heidi

En la siguiente ventana se deben seguir los siguientes pasos:

1- Ingresar el nombre absoluto, o relativo a la carpeta donde se encuentra la aplicación, oprimiendo el botón de examinar correspondiente. Ver Figura G.10

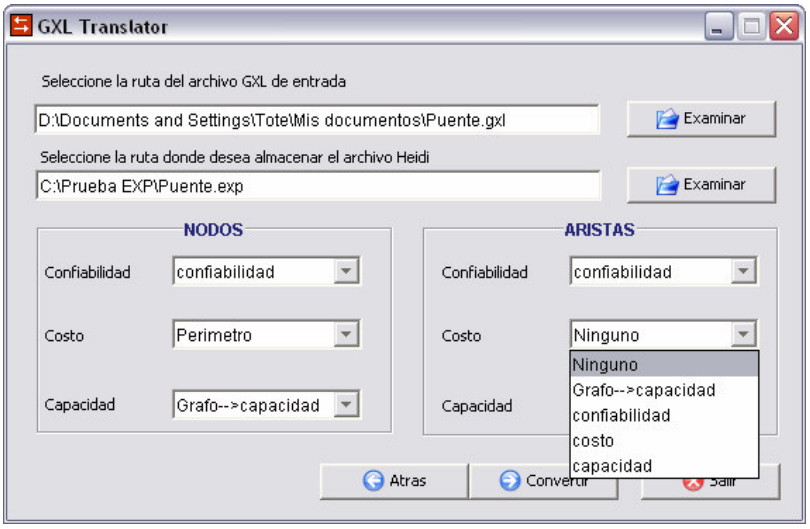


Fig. G.10: Ventana de conversión GXL a Heidi

2- La aplicación lee el archivo ingresado y se debe proceder a realizar el mapeo de los atributos del archivo GXL a los atributos Confiabilidad, Costo y Capacidad del archivo Heidi.

Todos los atributos de las aristas del archivo GXL, más todos los atributos del grafo, aparecen como disponibles para realizar el mapeo. Lo mismo sucede con los nodos. Si algún atributo no esta presente en determinado nodo o arista y es elegido para el mapeo entonces, en el lugar correspondiente del archivo Heidi, se generará un línea en blanco.

También aparece disponible la opción “Ninguno”, en caso que no se quiera realizar un mapeo hacia ese atributo del archivo Heidi.

Como se puede apreciar en la Figura G.10 los atributos a nivel del Grafo, aparecen disponibles para ser seleccionados para el mapeo y están distingos por el prefijo “Grafo→”

Si el archivo GXL seleccionado no es válido, se notifica el hecho con un mensaje análogo al de la Figura G.11

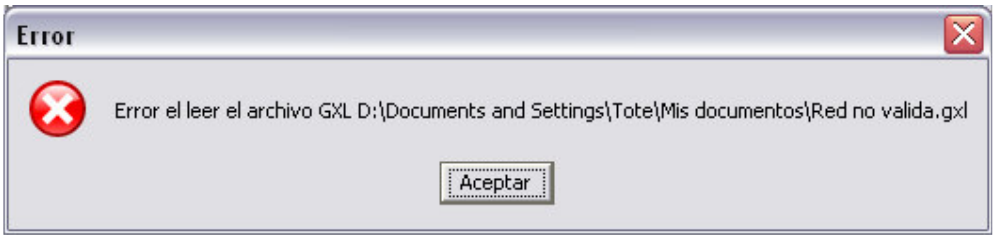


Fig. G.11: Error al leer archivo GXL

3- Ingresar el nombre absoluto o relativo a la carpeta donde se encuentra la aplicación, del archivo Heidi que se va a crear. Si el archivo ya existe también se puede oprimir el botón examinar correspondiente, para seleccionarlo. En caso que el archivo no tenga extensión .exp, la aplicación se la agregará automáticamente.

4- Finalmente oprimir el botón “Convertir”.

Si la conversión es exitosa, se desplegará un mensaje notificándolo. Ver Figura G.12

Fig. G.12: Confirmación de conversión de archivo Heidi

Si hubo algún error en la conversión se desplegará el mensaje de error correspondiente, por ejemplo, si el camino para el archivo Heidi a generar no es válido, se desplegará el mensaje de la Figura G.13.

Fig. G.13: No se puede generar archivo Heidi

En cualquiera de los dos casos, al clickear en botón Aceptar, vuelve a la ventana de traducción Heidi a GXL, para la siguiente conversión.

Y al igual que en traductor Heidi a GXL, en cualquier momento de la conversión se puede oprimir el botón “Anterior”, para volver a la ventana de selección de tipo de Conversión.

G.2. Manual de Usuario de GXL Viewer

G.2.1. Instalación y Ejecución

G.2.1.1. Requerimientos

Se requiere J2SE Java Runtime Environment (JRE) 1.4.0 o superior.

G.2.1.2. Instalación

Para realizar la instalación de GXL Viewer debe ejecutar el archivo llamado “Install.jar“. De esta manera se desplegará un asistente de instalación que lo asistirá en este proceso.

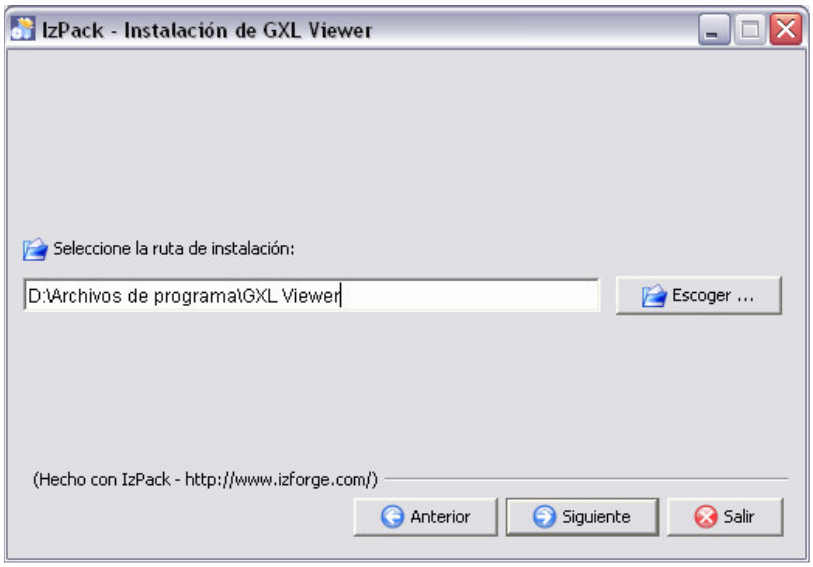


Fig. G.14: Selección ruta de instalación

Otra etapa, es la que pide que se seleccione los paquetes que desea instalar como se muestra en la Figura G.15. Dentro de los paquetes disponibles se encuentra

- Aplicación – paquete que contiene el ejecutable de la aplicación (la inclusión de este paquete es obligatoria)
- Documentación – manual de usuario y javadocs de la aplicación (la inclusión de este paquete es opcional)
- Fuentes – fuentes de la aplicación (la inclusión de este paquete es opcional)

Para incluir/excluir de la instalación a los paquetes opcionales se debe tildar/destildar el recuadro que aparece a la izquierda de cada uno de los paquetes.

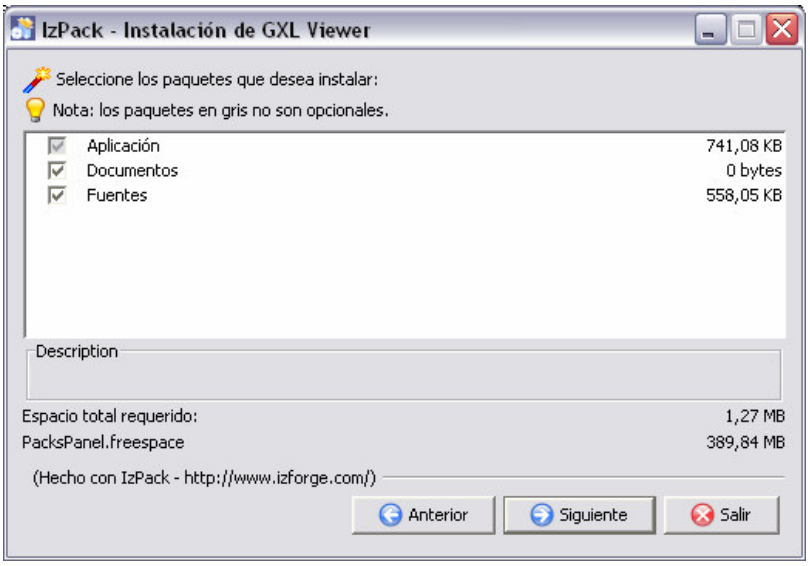


Fig. G.15: Selección de paquetes a instalar

G.2.1.3. Ejecución

Una vez instalada la aplicación la misma tendrá la estructura de carpetas que se muestra a continuación:

- Bin – contiene el archivo de extensión JAR que es el ejecutable de la aplicación
- Doc – contiene el manual de usuario y los javadocs de la aplicación. En caso de no instalarse el paquete de "Documentos" esta carpeta no aparecerá.
- Src – contiene los fuentes de la aplicación. En caso de no instalarse el paquete de "Fuentes" esta carpeta no aparecerá.
- Uninstaller – contiene un desinstalador de la aplicación

En la Figura G.16 muestra a modo de ejemplo como queda la estructura de la instalación si se selecciono como ruta de destino "D:\Archivos de Programa"

Fig. G.16: Estructura la instalación

Para ejecutar la herramienta debe posicionarse en la carpeta de nombre "bin" que se encuentra donde realizo la instalación y ejecutar el archivo llamado "GXL Viewer.jar"

G.2.2. Funcionalidades

G.2.2.1. Abrir archivo

Para abrir un archivo seleccione en el menú “Archivo” y luego la opción “Abrir” como se muestra en la Figura G.17.

Fig. G.17: Abrir Archivo

En caso de estar visualizando un grafo cuando realiza esta acción, éste se cerrará inmediatamente después de preguntarle si desea almacenar los cambios.

A continuación se despliega un selector de archivos para que elija el archivo con formato Graph eXchange Language (de aquí en adelanta nos referiremos como GXL) que desea visualizar.

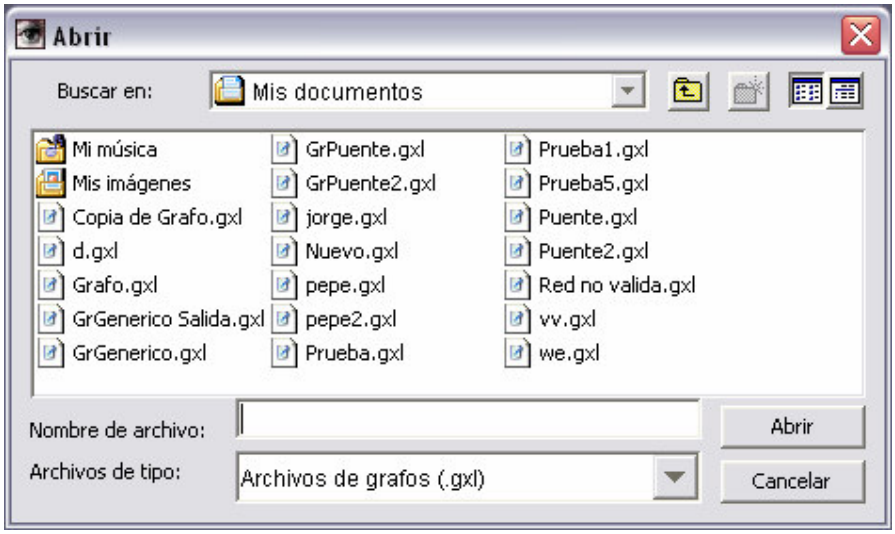


Fig. G.18: Selector para abrir archivos

Una vez seleccionado el archivo con el formato correcto el mismo será visualizado. En caso de que el archivo seleccionado tenga formato incorrecto se desplegará un mensaje de error.

G.2.2.2. Guardar archivo

Para guardar un archivo que se esta visualizando seleccione en el menú “Archivo” y luego la opción “Guardar” como se muestra en la Figura G.19. Esto provocara que se almacene toda la información con respecto a la visualización en el mismo archivo que se eligió visualizar.

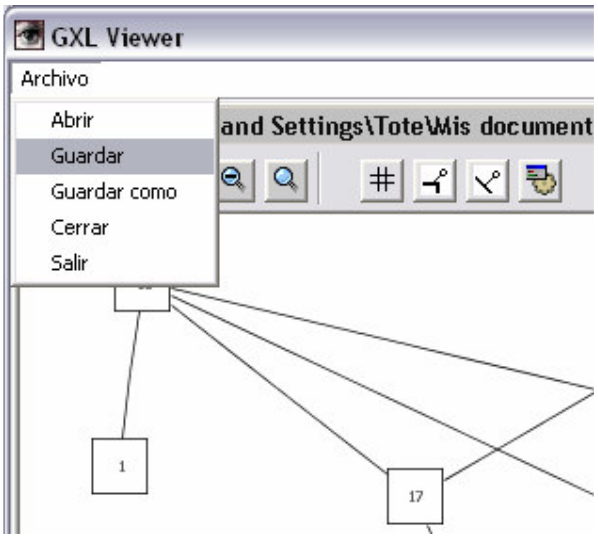


Fig. G.19: Guardar Archivo

G.2.2.3. Guardar archivo como

Para guardar un grafo en un lugar, o con un nombre distinto al grafo que se eligió visualizar, seleccione en el menú “Archivo” y luego la opción “Guardar como” esto se muestra en la Figura G.20.

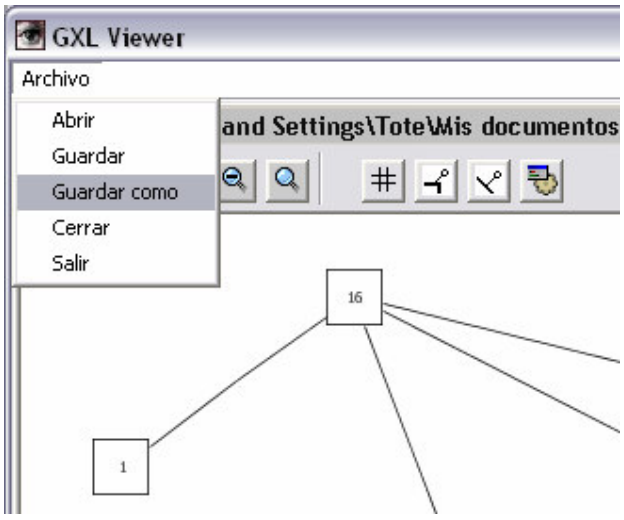


Fig. G.20: Guardar Archivo como

Una vez seleccionada esta opción se despliega un selector de archivos para que elija la ruta en donde se desea almacenar el grafo.

Fig. G.21: Selector para guardar archivos

Es importante destacar que una vez almacenado el grafo utilizando esta funcionalidad, la aplicación continua trabajando con el grafo seleccionado inicialmente.

G.2.2.4. Cerrar Archivo

Para cerrar un archivo que se esta visualizando seleccione en el menú “Archivo” y luego la opción “Cerrar” como se muestra en la Figura G.22.

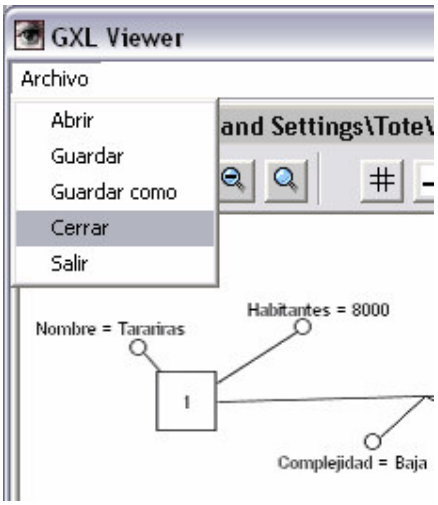


Fig. G.22: Cerrar archivo

Una vez seleccionada esta opción se desplegará un mensaje que le preguntara si usted desea almacenar los cambios realizados. En caso afirmativo presione “Si”, en caso contrario seleccione “No”.

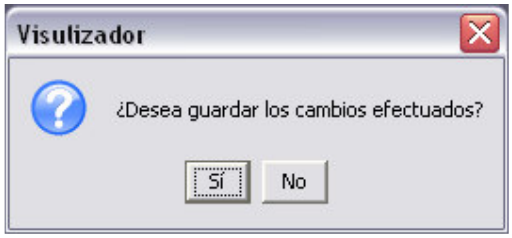


Fig. G.23: Dialogo de guardar cambios

G.2.2.5. Salir

Para salir de la aplicación seleccione en el menú “Archivo” y luego la opción “Salir” como se muestra en la Figura G.24.

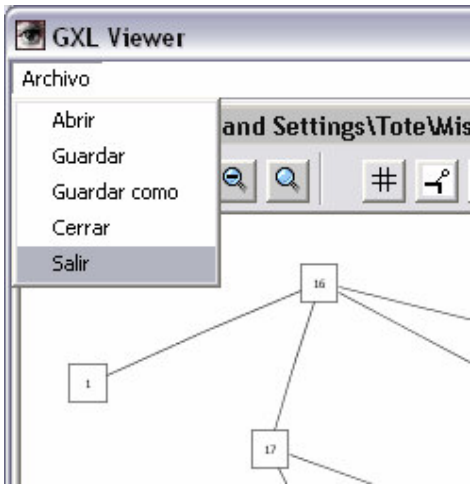


Fig. G.24: Salir

Una vez seleccionada esta opción y si usted estaba visualizando algún grafo, se desplegará un mensaje que le preguntara si usted desea almacenar los cambios realizados. En caso afirmativo presione “Si”, en caso contrario seleccione “No” (ver Figura G.23).

G.2.2.6. Ordenar del grafo

Haga clic sobre el nodo o atributo que desea mover. Si desea mover más de uno, seleccione los objetos teniendo apretada la tecla ctrl. o marque la superficie que contiene los nodos/atributos que desea seleccionar dejando presionado el botón izquierdo del mouse como se muestra en la Figura G.25. Al realizar esta operación todos los objetos seleccionados aparecerán con sus bordes en color rojo.

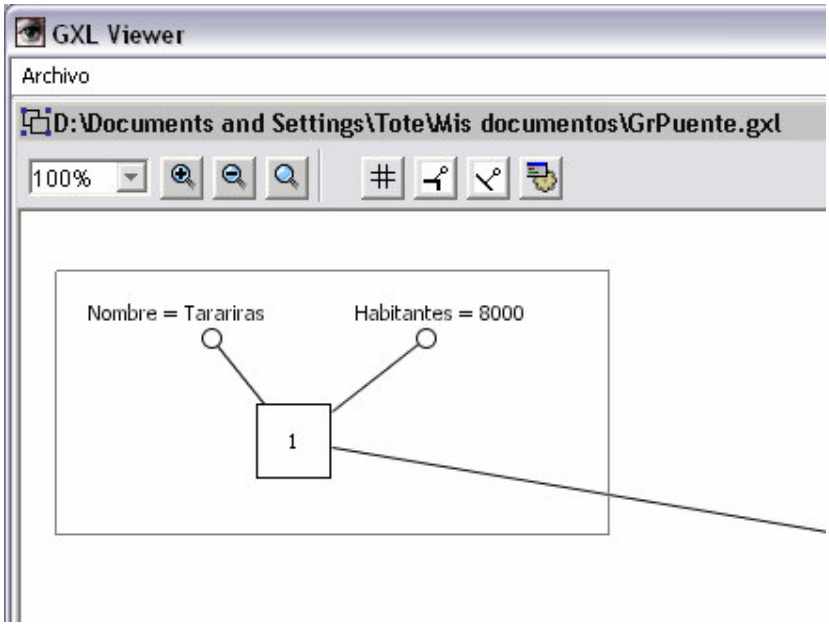


Fig. G.25: Selección múltiple

Presione el botón izquierdo del mouse sobre uno de los objetos seleccionados y arrástrelos hasta el lugar que los desea mover.

En el siguiente ejemplo moveremos hacia abajo, al nodo 1 y a sus dos atributos.

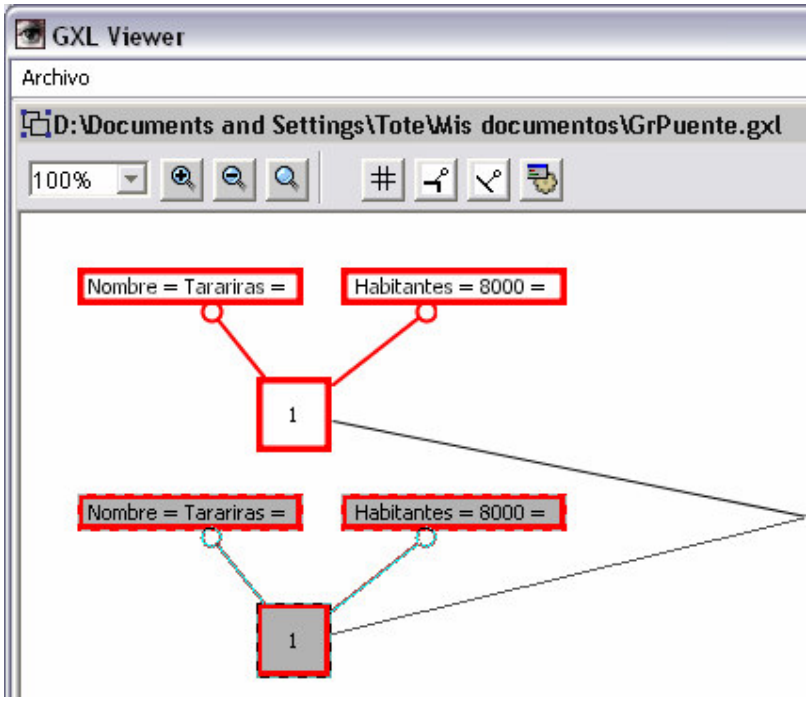


Fig. G.26: Selección múltiple

G.2.2.7. Zoom

La aplicación permite controlar el zoom con que se visualiza el grafo. Esto puede hacerse de varias maneras. Una de ellas es seleccionando un valor de la lista que se marca con rojo en la Figura G.27. o simplemente ingresando un valor en la misma.

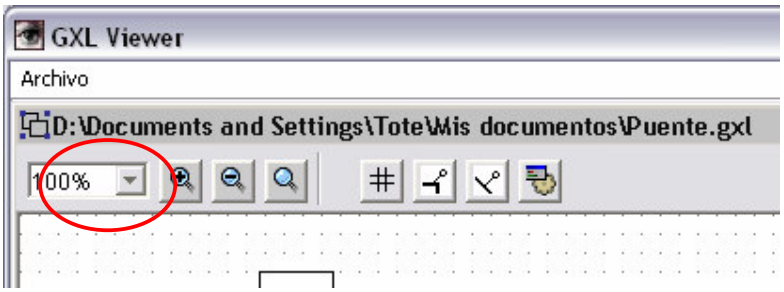


Fig. G.27: Zoom combo

Otra posibilidad es presionar los botones que aparecen marcados con rojo el la Figura G.28.; los cuales nos permiten (mirándolos de izquierda a derecha) aumentar en 25% el zoom actual, disminuir un 25% el zoom actual o poner el zoom en 100%.

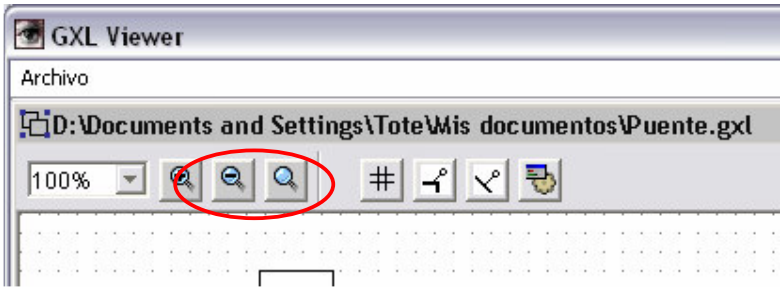


Fig. G.28: Zoom botones

G.2.2.8. Grilla

Esta funcionalidad se activa cuando se presiona el botón que aparece marcado con rojo en la Figura G.29.

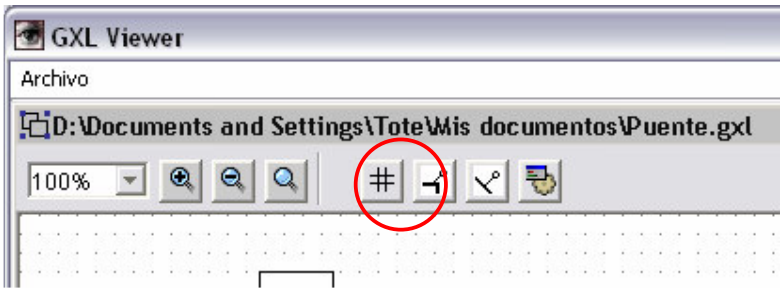


Fig. G.29: Grilla

G.2.2.9. Visualizar atributos

La aplicación permite visualizar u ocultar los atributos de un grafo, tanto sean atributos de la arista como de los nodos. Para visualizar/ocultar los atributos de los nodos debe presionar al botón de más a la izquierda que aparece en el recuadro de color rojo de la Figura G.30. Para visualizar/ocultar los atributos de las aristas debe presionar al botón de más a la derecha del mismo recuadro.

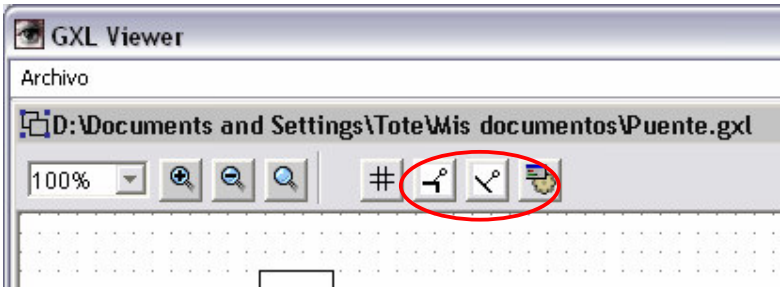


Fig. G.30: Atributos

G.2.2.10. Evaluar

Para activar la ventana de evaluación presione sobre el botón que aparece en el recuadro de color rojo de la Figura G.31. De esta manera la aplicación verificará si es posible evaluar dicho grafo verificando que todas las aristas del grafo posean un atributo de nombre “Confiabilidad”, en caso de ser posible se desliga la ventana que se muestra en la Figura G.26.

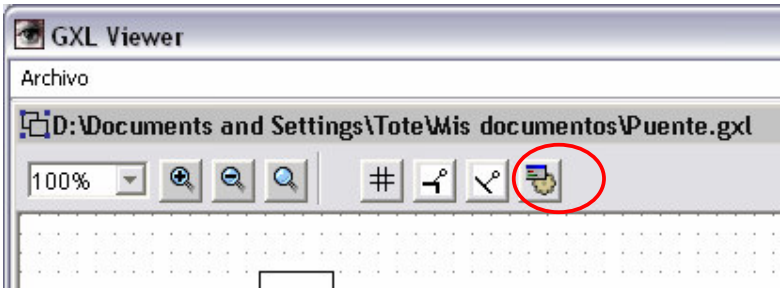


Fig. G.31: Evaluar

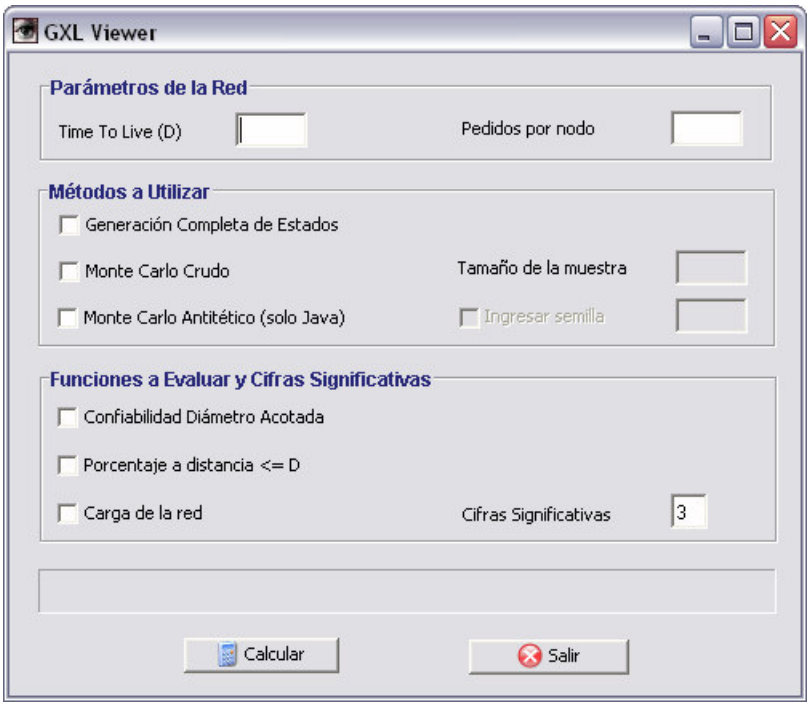


Fig. G.32: Ventana de evaluación de grafos

La evaluación se realiza como esta descrito en a la sección G.4 “Manual de Usuario del Evaluador de Grafos”, con la única diferencia, como se aprecia en la Figura G.32, de que aquí, no esta disponible la opción de correrlo en C++ y se omite el ingreso de la Cantidad de Nodos y la Cantidad de Conexiones por Nodo, dado que estos parámetros son para definir el grafo y en este caso el grafo que se evalúa es el que se esta visualizando.

G.3. Manual de Usuario de CDA Evaluator

G.3.1. Instalación y Ejecución

G.3.1.1. Requerimientos

Se requiere J2SE Java Runtime Environment (JRE) 1.4.0 o superior.

G.3.1.2. Instalación

Para realizar la instalación de CDA Evaluator debe ejecutar el archivo llamado “Install.jar“. De esta manera se desplegará un asistente de instalación que lo asistirá en este proceso.

Dentro del proceso de instalación se pueden destacar dos etapas:

Una etapa en la que se pide al usuario que seleccione la ruta de instalación (Figura G.33) pudiendo ingresar la misma posicionado el cursor en el cuadro de texto o presionando el botón de “Escoger” para seleccionarla.

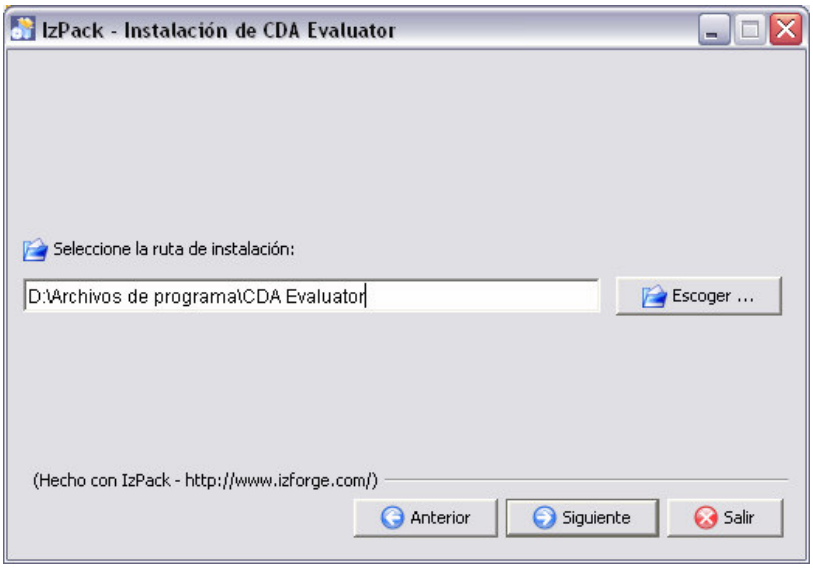


Fig. G.33: Selección ruta de instalación

Otra etapa, es la que pide que se seleccione los paquetes que desea instalar como se muestra en la Figura G.34. Dentro de los paquetes disponibles se encuentra

- Aplicación – paquete que contiene el ejecutable de la aplicación (la inclusión de este paquete es obligatoria)
- Documentación – manual de usuario y javadocs de la aplicación (la inclusión de este paquete es opcional)
- Fuentes – fuentes de la aplicación (la inclusión de este paquete es opcional)

Para incluir/excluir de la instalación a los paquetes opcionales se debe tildar/destildar el recuadro que aparece a la izquierda de cada uno de los paquetes.

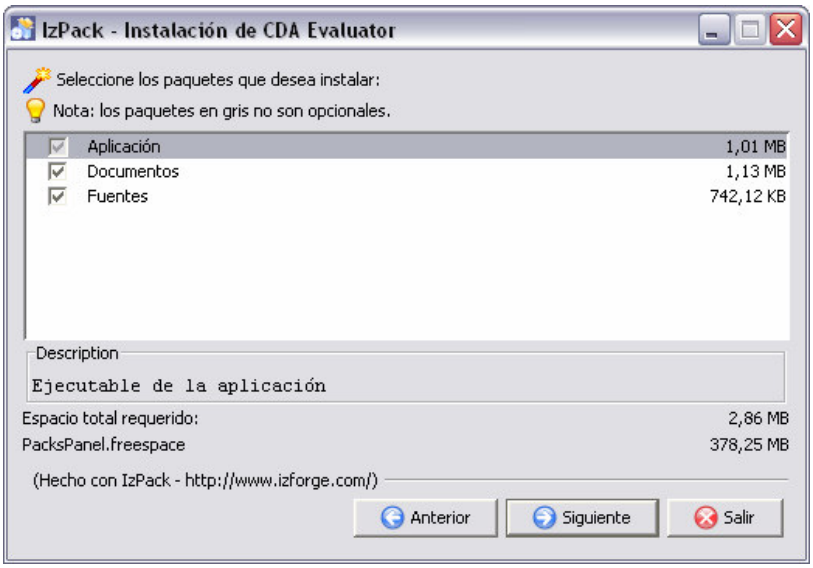


Fig. G.34: Selección de paquetes a instalar

Una vez finalizado el asistente de instalación la aplicación habrá quedado con la estructura de carpetas que se muestra a continuación:

- Bin – contiene el archivo de extensión JAR que es el ejecutable de la aplicación
- Doc – contiene el manual de usuario y los javadocs de la aplicación. En caso de no instalarse el paquete de “Documentos” esta carpeta no aparecerá.
- Src – contiene los fuentes de la aplicación. En caso de no instalarse el paquete de “Fuentes” esta carpeta no aparecerá
- Uninstaller – contiene un desinstalador de la aplicación

En la Figura G.35 muestra a modo de ejemplo como queda la estructura de la instalación si se selecciono como ruta de destino “D:\Archivos de Programa”

Fig. G.35: Estructura la instalación

El último paso necesario para culminar la instalación es poner en la variable de entorno “PATH” la carpeta “bin” que acaba de crear el asistente de instalación.

G.3.1.3. Ejecución

Para ejecutar la herramienta debe posicionarse en la carpeta de nombre "bin" que se encuentra donde realizo la instalación y ejecutar el archivo llamado "CDA Evaluator.jar"

G.3.2. Funcionalidades

En el presente documento se describe cada uno de los pasos necesarios para realizar una evaluación y posterior interpretación de los resultados. Para comenzar se muestra en la Figura G.36 un ejemplo de ejecución de la herramienta y posteriormente se explicarán cada una de las secciones (cuyo nombre aparece recuadrado en rojo) que se pueden distinguir en la figura.

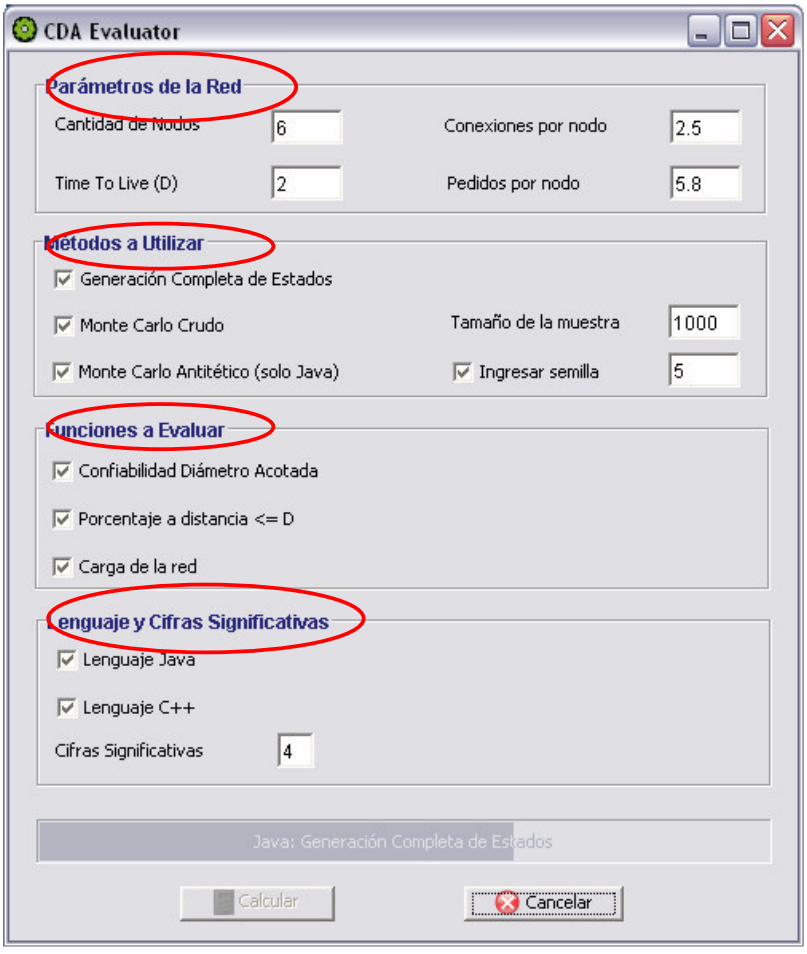


Fig. G.36: Ejemplo de evaluación de un grafo

G.3.2.1. Parámetros de la Red

Para comenzar se debe ingresar los parámetros de definen la red que se quiere evaluar.

- Cantidad de nodos de la red
- Conexiones por nodo, es un número real que representa la cantidad de conexiones promedio por nodo.
- Time-to-live, es la cantidad máxima de saltos que puede realizar un paquete.

- Pedidos por nodo, es un número real que representa la cantidad de pedidos promedio por nodo.

G.3.2.2. Métodos a Utilizar

En este punto el usuario debe seleccionar por lo menos uno de los tres métodos de cálculo que ofrece la herramienta.

- Generación Completa de estados
- Monte Carlo Crudo.
- Monte Carlo Antitético

Si se selecciona alguno de los métodos Monte Carlo se habilita para que obligatoriamente se ingrese el tamaño de la muestra que utilizará el método. Además si se desea se puede tildar la solapa de “Ingresar semilla” la cual habilita un campo para que se ingrese una semilla para generador de números aleatorios. Para este parámetro se debe ingresar valores enteros.

G.3.2.3. Funciones a Evaluar

Luego de seleccionar con que métodos se desea realizar el cálculo, es necesario elegir por lo menos una de las tres funciones a evaluar sobre la red.

- Confiabilidad Diámetro acotada
- Porcentaje
- Carga

G.3.2.4. Lenguaje y Cifras Significativas

Por último se debe seleccionar en que lenguaje se desea realizar el cálculo. Pudiéndose elegir Java, C++ u ambos. Cabe destacar que el método Monte Carlo Antitético fue implementado solo para Java.

La última de las opciones que permite seleccionar la aplicación es la cantidad de cifras significativas con que se despegará el resultado. Al comenzar esta opción aparece con un valor por defecto, igual a 3.

G.3.2.5. Calcular/Cancelar

Una vez definida las características de la red, los métodos a utilizar, las funciones a evaluar y el lenguaje en el que se realizará el cálculo, se debe presionar el botón de “Calcular“para iniciar la evaluación.

Si el lenguaje elegido para realizar la evaluación es Java, la misma podrá ser cancelada en cualquier momento presionando el botón de “Cancelar”, y se podrá monitorear el avance de los cálculos a través de una barra de progreso situada en la parte inferior de la pantalla de cálculo. El titulo de la barra de progreso denota el lenguaje y método sobre el cual se esta mostrando el progreso.

En el caso de ejecutarse en C++ la barra no muestra el progreso sino simplemente muestra en el titulo el lenguaje y el método que se esta calculando.

G.3.2.6. Resultados

Para desplegar los resultados de las evaluaciones seleccionadas se desplegarán un conjunto de tablas. Se muestran tantas tablas como combinaciones existan entre los lenguajes y los métodos previamente elegidos; pudiendo desplegar hasta 5 tablas, 3 para el lenguaje Java (uno por cada método a utilizado) y 2 para el lenguaje C++ (uno por cada método utilizado).

Cada una de las tablas desplegadas posee una fila por cada una de las funciones a evaluar que fueron previamente elegidas.

La cantidad de columnas y el significado de alguna de ellas, varia dependiendo si la tabla muestra los resultados de evaluar un método exacto o uno aproximado.

En la Figura G.37 se muestra a modo de ejemplo como se despliegan los resultados para una corrida del método exacto “Generación Completa de Estados”. Los parámetros seleccionados para esta corrida son los que aparecen en la Figura G.36

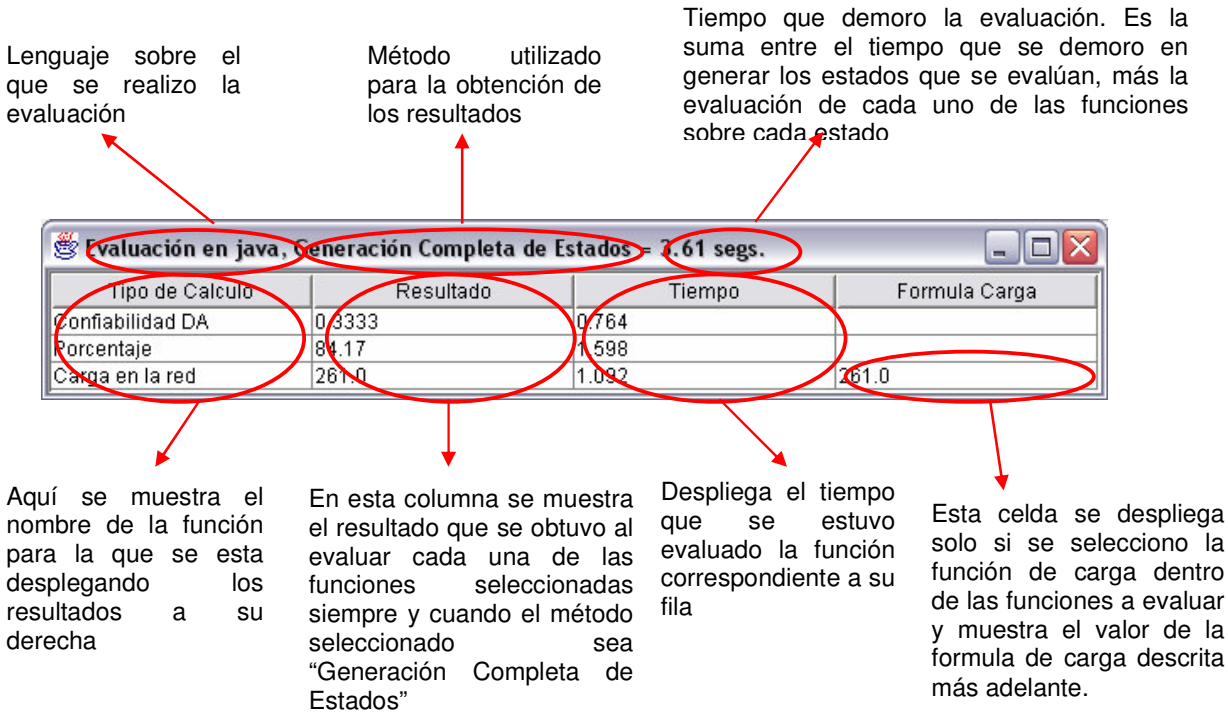


Fig. G.37: Ejemplo de resultado de evaluación utilizando un método exacto

La función de carga es una estimación de la misma dada por la siguiente fórmula:

$$carga=NP(CCP(N-1)(1-CCP^{D}(1-1/(N-1))^{D})/(N-1)-CCP(N-2))$$

siendo N la cantidad de nodos, P la cantidad de pedidos por Nodo y CCP la cantidad de conexiones promedio por nodo.

En la Figura G.38 veremos como se despliegan los resultados para una corrida del método “Monte Carlo Crudo” (coincide con la forma en que se despliegan los datos para el “Monte Carlo Antitético”). Los parámetros seleccionados para esta corrida son los que aparecen en la Figura G.36. Aquí se explicará solamente los puntos en los que difiere con respecto a la tabla resultado del método “Generación completa de estados”.

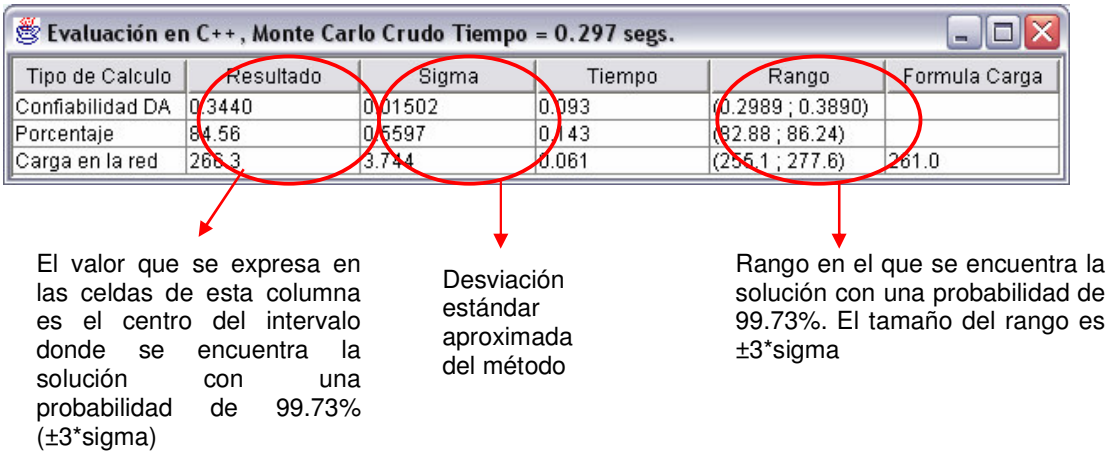


Fig. G.38: Ejemplo de resultado de evaluación utilizando Monte Carlo

Nota: Para expresar un número real se utiliza el punto como separador entre la parte entera y la real.

G.4. Manual de Usuario de CDA Optimizer

G.4.1. Instalación y Ejecución

G.4.1.1. Requerimientos

Se requiere J2SE Java Runtime Environment (JRE) 1.4.0 o superior.

G.4.1.2. Instalación

Para realizar la instalación de GXL Optimizer debe ejecutar el archivo llamado “Install.jar“. De esta manera se desplegará un asistente de instalación que lo asistirá en este proceso.

Dentro del proceso de instalación se pueden destacar dos etapas:

Una etapa en la que se pide al usuario que seleccione la ruta de instalación (Figura G.39) pudiendo ingresar la misma posicionado el cursor en el cuadro de texto o presionando el botón de “Escoger” para seleccionarla.

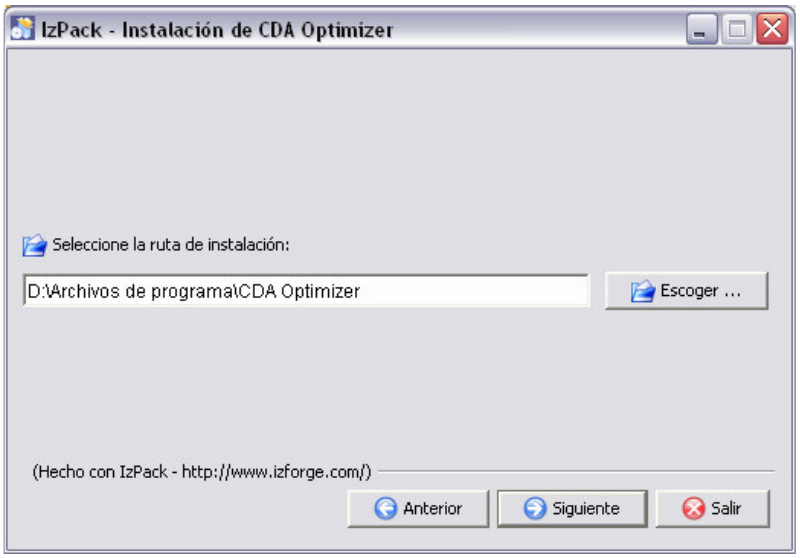


Fig. G.39: Selección ruta de instalación

Otra etapa, es la que pide que se seleccione los paquetes que desea instalar como se muestra en la Figura G.40. Dentro de los paquetes disponibles se encuentra

- Aplicación – paquete que contiene el ejecutable de la aplicación (la inclusión de este paquete es obligatoria)
- Documentación – manual de usuario y javadocs de la aplicación (la inclusión de este paquete es opcional)
- Fuentes – fuentes de la aplicación (la inclusión de este paquete es opcional)

Para incluir/excluir de la instalación a los paquetes opcionales se debe tildar/destildar el recuadro que aparece a la izquierda de cada uno de los paquetes.

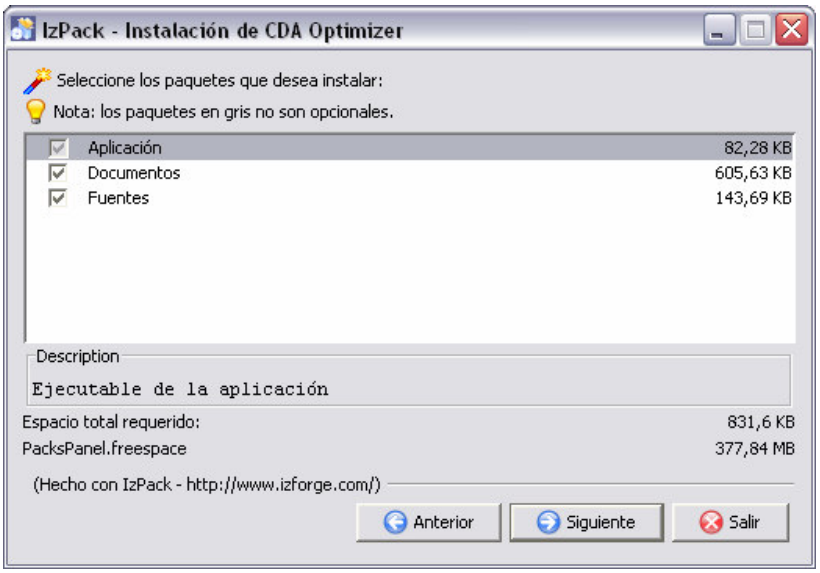


Fig. G.40: Selección de paquetes a instalar

G.4.1.3. Ejecución

Una vez instalada la aplicación la misma tendrá la estructura de carpetas que se muestra a continuación:

- Bin – contiene el archivo de extensión JAR que es el ejecutable de la aplicación
- Doc – contiene el manual de usuario y los javadocs de la aplicación. En caso de no instalarse el paquete de “Documentos” esta carpeta no aparecerá.
- Src – contiene los fuentes de la aplicación. En caso de no instalarse el paquete de “Fuentes” esta carpeta no aparecerá
- Uninstaller – contiene un desinstalador de la aplicación

En la Figura G.41 muestra a modo de ejemplo como queda la estructura de la instalación si se selecciono como ruta de destino “D:\Archivos de Programa”

Fig. G.41: Estructura la instalación

Para ejecutar la herramienta debe posicionarse en la carpeta de nombre “bin” que se encuentra donde realizo la instalación y ejecutar el archivo llamado “GXL Optimizer.jar”

G.4.2. Funcionalidades

En esta sección del anexo se describe cada uno de los pasos necesarios para realizar una optimización y posterior interpretación de los resultados. Al ejecutar el optimizador aparecerá una ventana como la que se muestra en la Figura G.42. En ella se identifica con un ovalo rojo las secciones que se deben ingresar para realizar la optimización, las cuales se describen a continuación.

G.4.2.1. Parámetros de la Red

Para comenzar se debe ingresar los parámetros de definen la red que se quiere evaluar.

- Cantidad de nodos de la red
- Pedidos por nodo, es un número real que representa la cantidad de pedidos promedio que un nodo realiza en una unidad de tiempo.

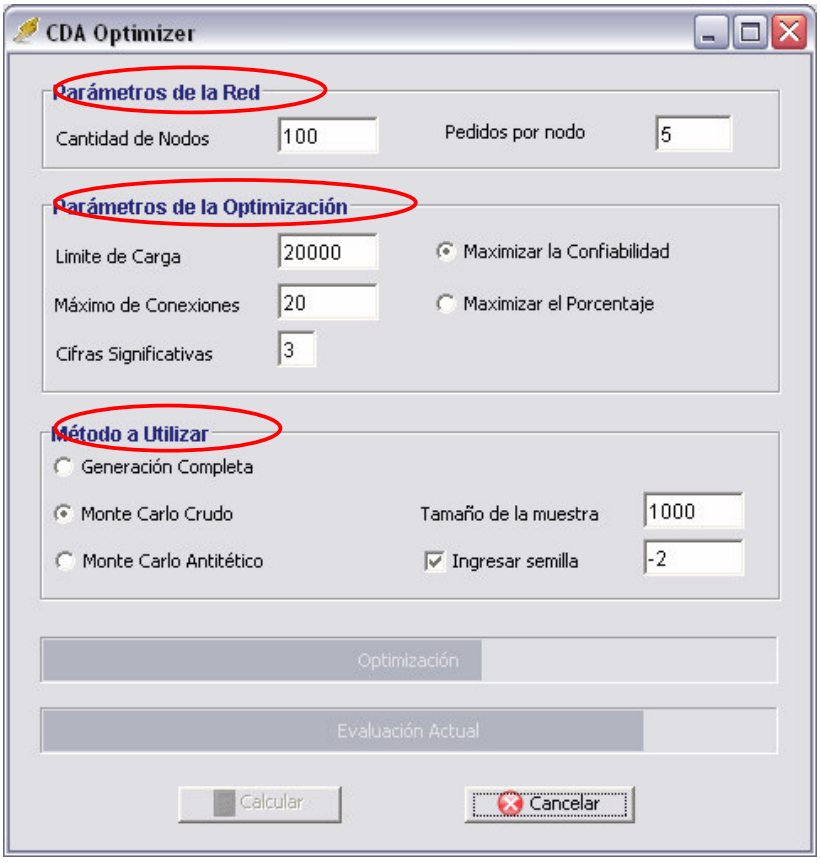


Fig. G.42: Ejemplo de optimización de una red

G.4.2.2. Parámetros de la Optimización

Límite de carga: es un entero que representa la máxima cantidad de paquetes que pueden admitirse ser enviados en la red, en una unidad de tiempo.

Máximo de conexiones: es un número entero que representa la cantidad de conexiones promedio por nodo máxima que puede tener la red, para ser una solución factible en la optimización. Si no se especifica el valor por defecto que toma este parámetro es N-2. El caso

con CCP=N-1, no se considera ya que de hacerlo la solución es trivial y sería el punto (N-1,1), siempre que este punto cumpliera la restricción de carga, por haberse definido la confiabilidad de las aristas como el conciente en entre la CCP y N-1.

Además de ingresar los parámetros antes seleccionados es requerido elegir cual es la función que se desea maximizar sujeto a las restricciones ya planteadas. Para realizar esta decisión se debe elegir entre maximizar la: confiabilidad diámetro acotada ó el porcentaje de paquetes de nodos que se encuentran a distancia menor que TTL (D), tildando el recuadro que aparece a la izquierda de función que desea maximizar.

La última opción de esta sección de la aplicación, permite seleccionar la cantidad de cifras significativas con que se despegará el resultado. El valor por defecto inicial de este campo es 3.

G.4.2.3. Métodos a utilizar

En este punto el usuario debe seleccionar por uno de los tres métodos de cálculo que ofrece la herramienta.

- Generación Completa de estados
- Monte Carlo Crudo.
- Monte Carlo Antitético

Si se selecciona alguno de los métodos Monte Carlo se habilita para que obligatoriamente se ingrese el tamaño de la muestra que utilizará el método. Además si se desea se puede tildar el checkbox “Ingresar semilla” el cual habilita un campo para que se ingrese una semilla para generador de números aleatorios. Para este parámetro se debe ingresar valores enteros.

Se recomienda para redes mayores a 7 nodos, utilizar un método Monte Carlo y no la Generación Completa de Estados, por el tiempo que lleva cada evaluación con este método.

G.4.2.4. Calcular/Cancelar

Una vez definida las características de la red, los parámetros de la optimización y el método a utilizar se debe presionar el botón de “Calcular” para iniciar la optimización.

En el transcurso de la optimización la misma podrá ser cancelada en cualquier momento presionando el botón de “Cancelar”, y se podrá monitorear el avance de los cálculos a través de dos barras de progreso situadas en la parte inferior de la pantalla de cálculo. En la barra superior se puede observar el avance de la optimización en general, mientras que la barra inferior muestra el progreso de la evolución actual.

Esta funcionalidad, de poder ver el grado de avance del calculo y la posibilidad de cancelarlo es de suma utilidad, cuando se quiere experimentar con unos determinados parámetros y no se tiene una estimación previa de cuanto puede demorar, ya que al ver el grado de avance el usuario puede decidir si desea continuar o no. Tener en cuenta que las primeras evoluciones demoran más que las últimas (esto se debe a que hacen en orden decrecientes de C).

G.4.2.5. Resultados

La aplicación devuelve en pantalla, un mensaje indicando para que valores de Cantidad de Conexiones promedio por Nodo y Diámetro (TTL) se obtiene el mayor valor de Confiabilidad Diámetro Acotada o Porcentaje, según se haya elegido una función objetivo o la otra, cuanto es ese valor y la carga que se obtendría configurando para red a trabajar con esos valores de Conexiones y Diámetro. La Figura G.43 muestra dicho mensaje para el problema instanciado con los valores de la Figura G.42.

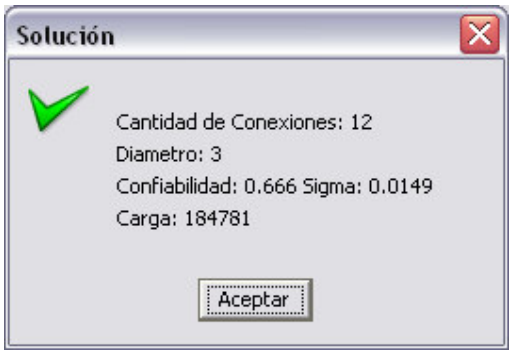


Fig. G.43: Mensaje indicando la solución al problema.

Puede suceder que para la restricción de carga que se ingresa, el óptimo no exista. Esto sucede cuando para todos las duplas del espacio de soluciones, que son candidatas, es decir

que cumplen con la restricción de carga, se tiene una Confiabilidad Diámetro Acotada o Porcentaje de valor 0. Esto sucede si se ingresa un límite que no sea lo suficientemente grande. En este caso se desplegará el mensaje que muestra la Figura G.44

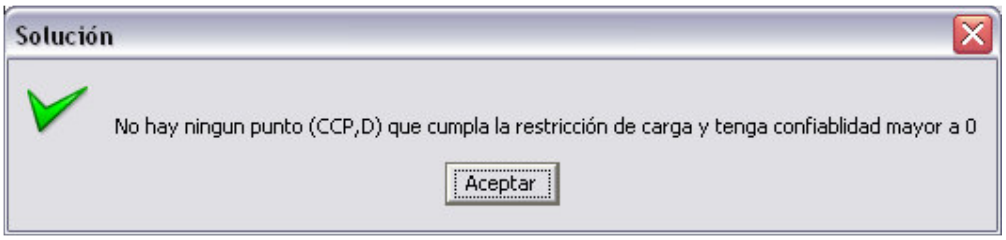


Fig. G.44: Mensaje indicando que no existe solución al problema.

En ambos casos se genera un archivo de nombre frontera.txt, en el directorio donde reside la aplicación, el cual registra en sus primeras líneas los parámetros de la ejecución y luego para cada punto de la frontera despliega el valor de la función objetivo y la carga en dicho punto.

Si pensamos la solución como una dupla (CCP,D) entonces podemos ver el espacio de soluciones como una matriz, en la cual se desea encontrar el punto que, cumpliendo la condición de carga maximiza la función objetivo seleccionada. En la Figura G.45, por ejemplo, se muestra un espacio de soluciones para N=20 y LC=5.000. En gris claro se marcan los puntos que cumplen la restricción de carga, en gris oscuro los que además pertenecen a la frontera y en negro el punto donde se da el óptimo. Informalmente un punto pertenece a la frontera, si el cumple la restricción de carga y todo otro punto cuya Cantidad de Conexiones o Diámetro es mayor, no la cumple.

CP \ D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	20	39	57	74	90	105	120	133	146	159	170	181	192	202	211	220	228	236	244
2	40	116	259	531	1.047	2.024	3.875	7.381	14.026	26.615	50.468	95.664	#####	#####	#####	#####	#####	#####	#####
3	60	231	715	2.093	6.007	17.134	48.756	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
4	80	383	1.532	5.885	22.382	84.897	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
5	100	574	2.817	13.446	63.791	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
6	120	802	4.679	26.718	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
7	140	1.068	7.225	48.055	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
8	160	1.373	10.563	80.217	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
9	180	1.715	14.800	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
10	200	2.095	20.045	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
11	220	2.513	26.404	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
12	240	2.968	33.986	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
13	260	3.462	42.899	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
14	280	3.994	53.249	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
15	300	4.563	65.145	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
16	320	5.171	78.694	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
17	340	5.816	94.005	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####
18	360	6.499	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####	#####

Fig. G.45: Espacio de Soluciones para N=20 y LC=5.00

Cabe destacar que en esta versión al no poderse seleccionar el nombre y la ubicación del archivo con los datos de la frontera, si la aplicación se ejecuta desde una unidad de CD Rom o en general desde una carpeta en la que no se tiene derechos de escritura, el archivo frontera.txt no se podrá generar. Esta situación, de darse, es indicada por mensaje correspondiente.

A continuación, se muestra el archivo frontera.txt para el caso indicado en la Figura G.42

```
N = 100
Limite de Carga= 200000
Pedidos = 1
Valor máximo de Conexiones = 20
Metodo = Monte Carlo Crudo
Tamaño de la muestra = 1000
Semilla = 1


Cantidad de Conexiones: 20
Diametro: 2
Confiabilidad: 0.0 Sigma: 0.0
Carga: 41595

Cantidad de Conexiones: 12
Diametro: 3
Confiabilidad: 0.658 Sigma: 0.0150
Carga: 184781

Cantidad de Conexiones: 6
Diametro: 4
Confiabilidad: 0.0070 Sigma: 0.00263
Carga: 151041
```

```
Cantidad de Conexiones: 4
Diametro: 5
Confiabilidad: 0.0 Sigma: 0.0
Carga: 131412

Cantidad de Conexiones: 3
Diametro: 6
Confiabilidad: 0.0 Sigma: 0.0
Carga: 104318

Cantidad de Conexiones: 2
Diametro: 10
Confiabilidad: 0.0 Sigma: 0.0
Carga: 188639

Cantidad de Conexiones: 1
Diametro: 99
Confiabilidad: 0.0 Sigma: 0.0
Carga: 6276
```

Fig. G.45: Frontera.txt para N=20, LC=5.00, CDA,

Como se puede apreciar el óptimo ocurre en CCP=12 y Diámetro=3, tal como lo muestra la Figura G.43.

G.5. Acceso web a las aplicaciones

Todas las aplicaciones están accesibles en la web, a través de la página de difusión de resultados de nuestro proyecto de grado, que es la siguiente:
http://www.fing.edu.uy/inco/grupos/invop/cda2005

En la misma se puede encontrar: una descripción del proyecto, la posibilidad de ejecutar con Java Web Star, cada una de las aplicaciones desarrollada, además de estar disponibles para bajar, el informe final del proyecto, la presentación que se realizara del mismo ante el tribunal, y las planillas con los pruebas de validación y análisis de resultados, junto con algunos archivos Heidi y GXL, que se usaron durante el testing del Visualizador y el Traductor de grafos.

Para poder ejecutar las aplicaciones desde la web es requerido que el cliente tenga instalado, la Java Runtime Enviroment, versión 1.4.1 o superior y Java Web Start versión 1.2 o posterior, el cual esta disponible en: http://java.sun.com/products/javawebstart/

La tecnología Java Web Start es una solución para el deployment de aplicaciones la través de la Web (por más información acerca de la misma recomendamos leer el siguiente link http://java.sun.com/products/javawebstart/docs/readme.html). Al hacer click en el link de la aplicación, como muestra la Figura G.46, la aplicación comienza a descargarse como se muestra en la Figura G.47.

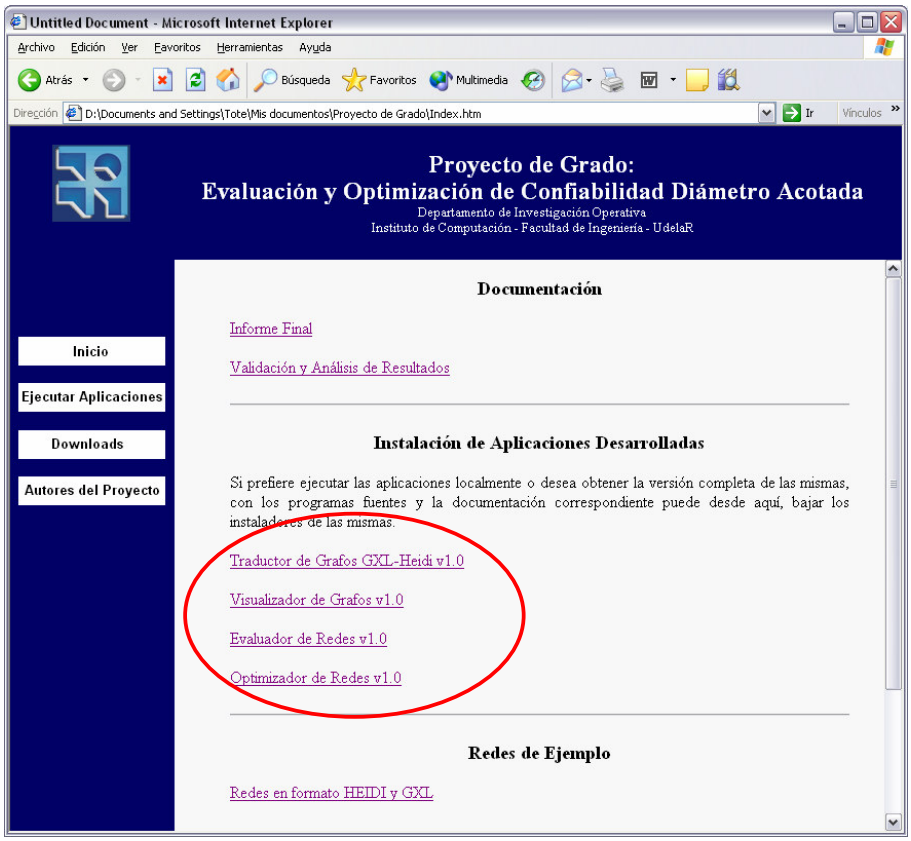


Fig. G.46: Página de difusión de los resultados

Las aplicaciones desarrolladas, a excepción del evaluador, necesitan acceder a sistema de archivos. Por ejemplo el visualizador necesita acceder a un archivo GXL para leerlo y eventualmente grabarlo. Es por dicha razón que los archivos .jar debieron ser firmados digitalmente. Al estar firmados las aplicación pueden tener un acceso sin restricciones al sistema, dado que el usuario lo esta permitiendo al aceptar la firma digital de la aplicación.

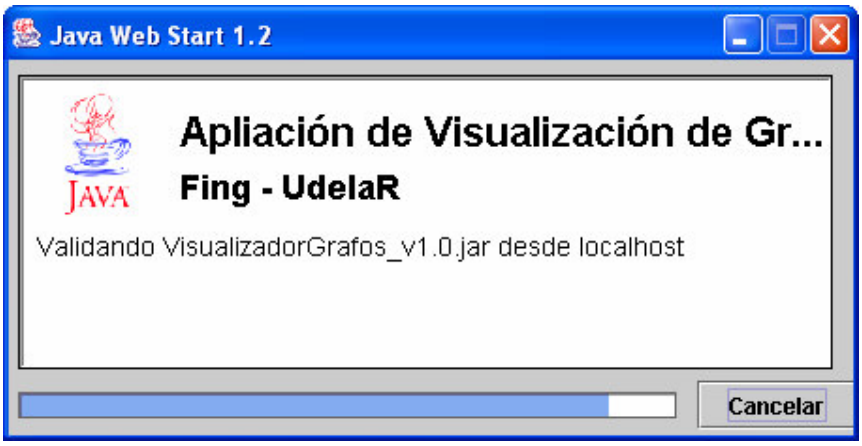


Fig. G.47: Descarga de la aplicación publicada.

Por lo que luego de descargada la aplicación el usuario recibirá un mensaje similar al de la Figura G.48, el cual debe aceptar para poder ejecutar la aplicación.

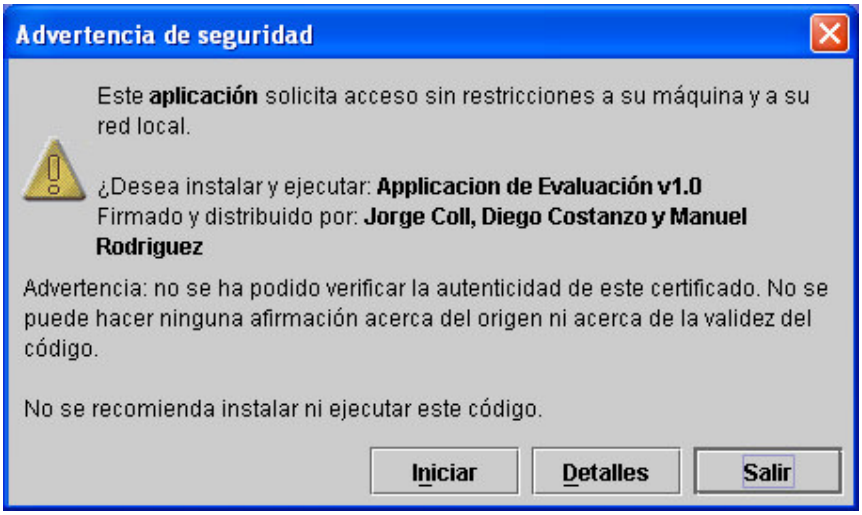


Fig. G.48: Confirmación de aceptación del jar firmado digitalmente.

Al hacer click en el link de la aplicación, el explorador invoca a Java Web Start, el cual baja el archivo JAR, lo cachea y ejecuta la aplicación. Todo este proceso es automático, y transparente para el usuario. Luego la aplicación queda del lado del cliente y puede ser ejecutada, desde un acceso directo que Java Web Start puede crear en el escritorio, desde el propio Java Web Star, como muestra la Figura G.49 o volviendo a clickear en el link de la página del proyecto. A su vez si hay una nueva versión disponible esta se actualiza en forma automática y transparente.

Fig. G.49: Ejecución de una aplicación desde Java Web Start.

:: Apéndice H

:: Gestión del Proyecto

H.1. Cronograma del Proyecto

Inicialmente se definió el cronograma que se muestra en la figura H.1 para al realización del proyecto. En el básicamente se estimaba un proyecto de 8 meses, iniciando en la segunda quincena de abril y terminado en la segunda quince de Diciembre.

Se distinguieron 3 fases del proyecto: el estudio de la representación de grafos y el desarrollo de las herramientas de visualización y traducción de Grafos, el estudio, la implementación y el análisis de la evaluación de la Confiabilidad Diámetro Acotada y una tercer fase para el estudio, implementación y el análisis de la Optimización. Se habían asignado unos dos meses para cada fase del proyecto y los dos meses finales para la preparación del informe final y la presentación.

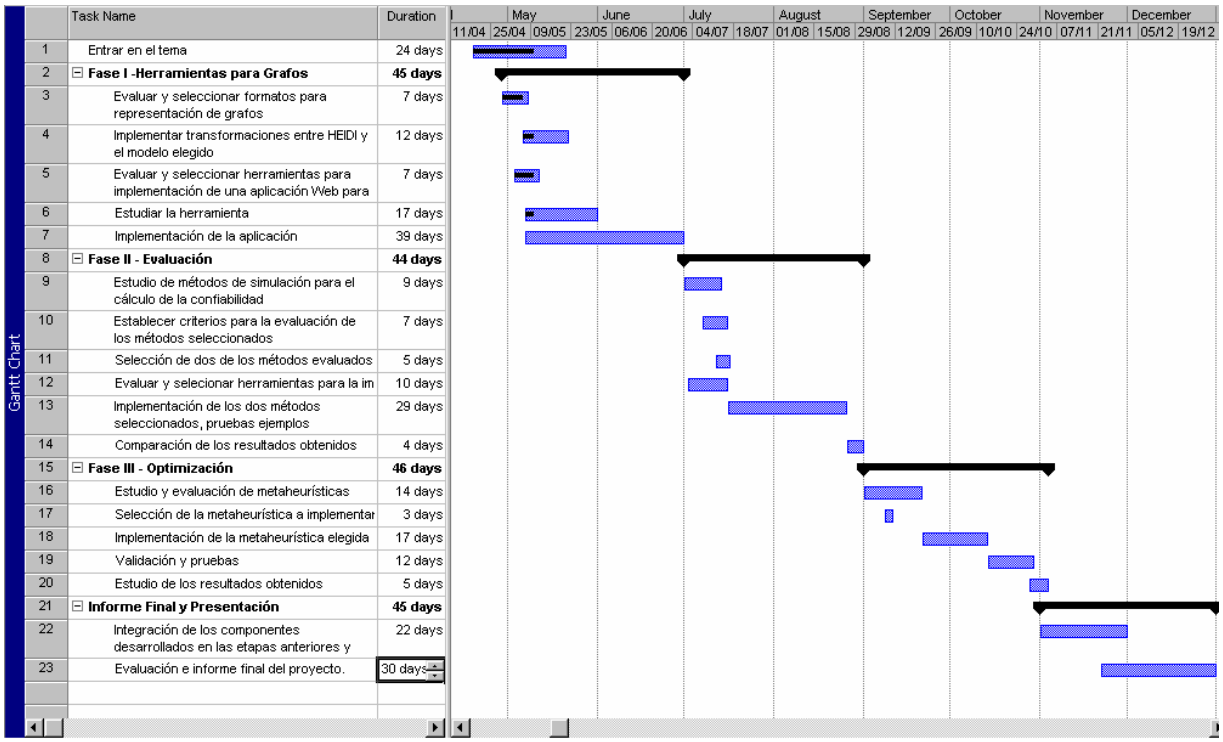


Fig. H.1: Cronograma definido inicialmente

Sin embargo, el cronograma llevado acabo es el que se muestra en la figura H.2. Como se puede apreciar algunas tareas llevaron más tiempo del esperado como por ejemplo el desarrollo del visualizador o la validación y análisis de resultados de la evaluación. En Enero y Febrero el proyecto estuvo en standby, retomándolo a principios de marzo, para su culminación a fines de Abril. En estos dos últimos meses se hizo el informe final, a excepción de los Anexos de representación de grafos, GXL y la documentación del diseño de la aplicaciones que se hicieron el correr del año pasado, y en paralelo se mejoraron las aplicaciones, haciéndose las versiones web, los setup, los manuales de usuario y la página del proyecto

En la figura H.2 se puede apreciar el cronograma de las dos primeras fases.

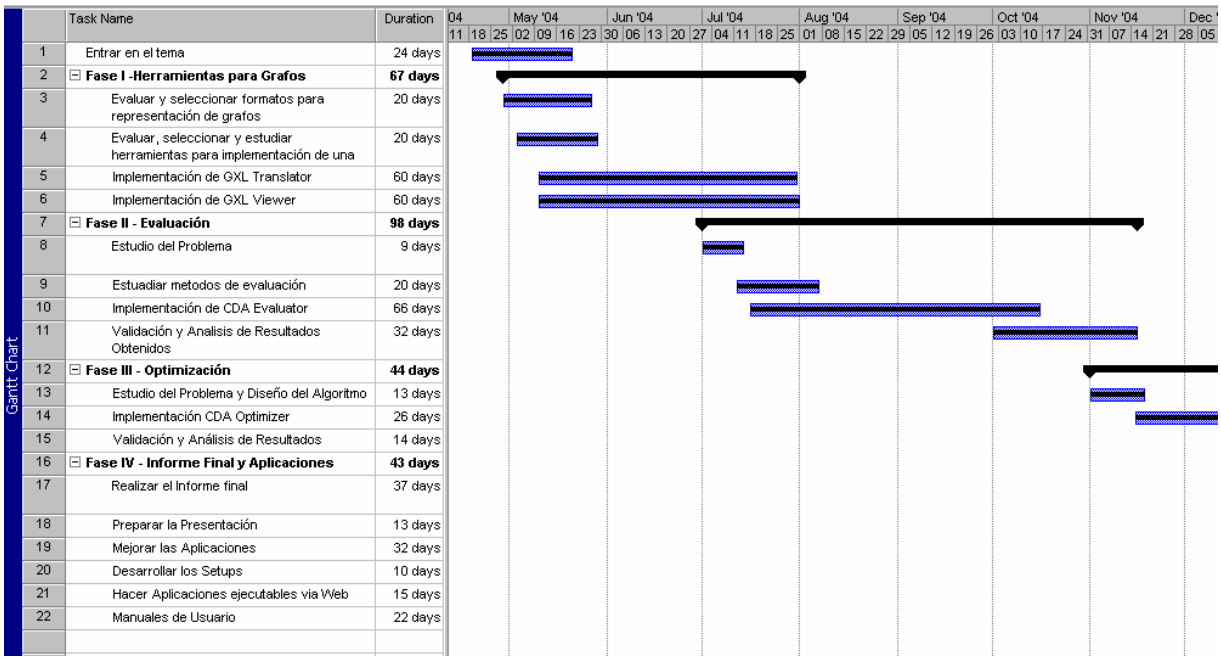


Fig. H.2: Cronograma llevado acabo de Fases I y II

En la figura H.3 se muestra el cronograma de las dos últimas fases.

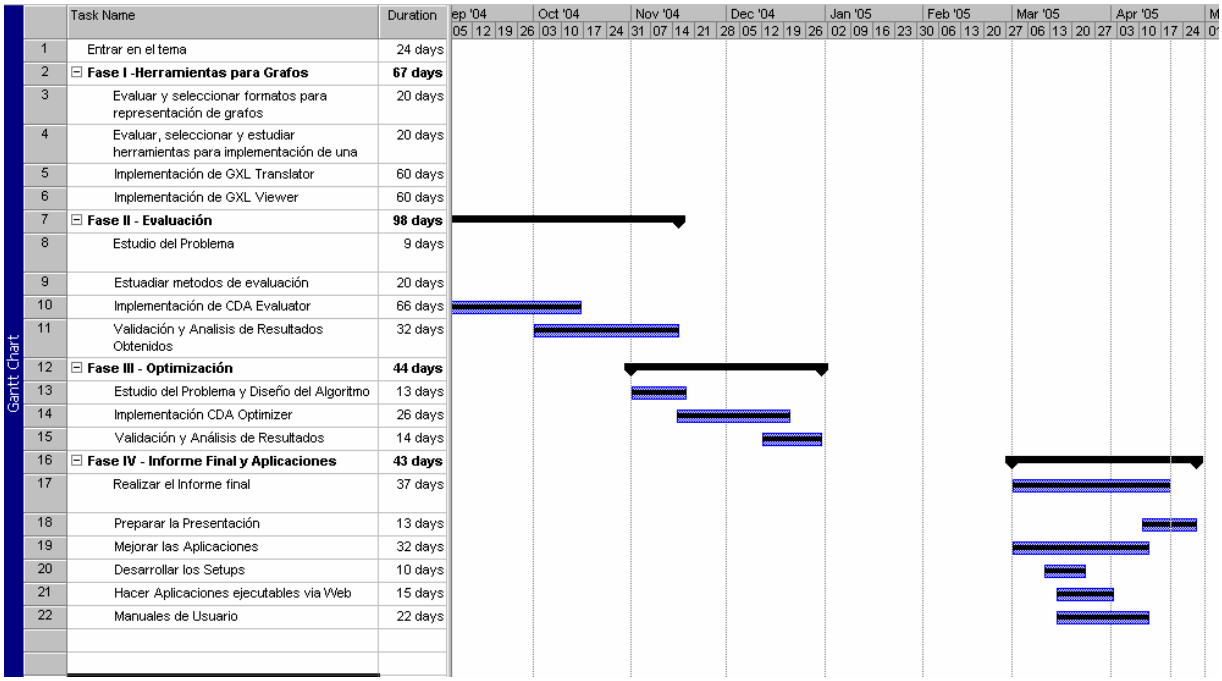


Fig. H.3: Cronograma llevado acabo de Fases III y IV

H.2. Actas de las reuniones

Minuta de Reunión #1 - 26/04/2004 Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Lunes 26 de Abril de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Manuel Rodriguez	Miembro del Grupo
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Tener unan visión más clara del proyecto y del modelo conceptual
- Encaminar la parte de representación de grafos.
- Determinar horarios y frecuencia de reuniones
- Definición de tareas para la próxima reunión de avance.

Desarrollo de Temas:

Tener una visión mas clara del proyecto y del modelo conceptual

La idea del proyecto, es implementar uno dos algoritmos de cálculo de confiabilidad y una meta heurística para la optimización de redes con el modelo de la confiabilidad diámetro acotada. Para ello se reutilizara (y eventualmente escribirá) software base disponible.

Se comentaron los posibles modelos de optimación que surgen de la conjugación de 3 variables, la confiabilidad, el costo y el tráfico. Las 3 combinaciones posibles son:

- Maximizar la confiabilidad sujeto al restricciones de costo y tráfico
- Minimizar el costo sujeto a condiciones de confiabilidad o tráfico y finalmente
- Minimizar el tráfico sujeto a la confiabilidad y el costo.

A su vez también se podría hacer una optimización múltiple objetivo, la cual consiste en definir una curva optima, en el espacio definido por cada una de las variables.

A su vez, si hizo hincapié en el hecho de que en el momento de definir el modelo, se debe tener en mente la utilización de este para la redes P2P. En este sentido por ejemplo nuestro modelo será para K=V.

Estas definiciones sobre el modelo se irán haciendo en las próximas semanas, según las necesidades del proyecto marco donde se inscribe el nuestro.

Tareas:

- Continuar leyendo trabajos relacionados en el área
- Repasar métodos de simulación, especialmente Monte Carlo
- Estudiar meta heurísticas, como por ejemplo Algoritmos evolutivos y Tabú Search.

Encaminar la parte de representación de grafos

En esta área se sugiere trabajar en la siguiente dirección:

- Leer el trabajo de Pablo Rodríguez Bocca “Lenguajes canónicos para la descripción de Grafos” publicado en la biblioteca del Instituto.

- Ver el estado del arte en el tema dado que el documento tiene mas de un año.

- Seleccionar un lenguaje.

- Ver si existe algún Plug-in o Applet de java para visualizar un grafo en XML y finalmente

- Convertir desde Heide al lenguaje elegido.

Determinar Horarios y frecuencia de reuniones

Se fijo dijo una reunión de avance con el tutor los días Jueves, en principio a las 20 hs, con frecuencia quincenal, comenzando el 6 de mayo de 2004.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión de avance avanzar en las siguientes:
Tareas:

- Armar un cronograma tentativo
- Definir un lenguaje para la representación de grafos para validarlo con el tutor.

Tareas en curso:

Tarea	Status (%)	Comentarios
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	90	Falta aprobación del tutor
Definir un cronograma tentativo	70	Falta aprobación del Tutor y mejorar la representación grafica (que se pueda expandir)
Continuar leyendo trabajos relacionados al área.	20	
Repasar métodos de simulación (Monte Carlo y otros)	0	
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	
Evaluar alguna técnica para el manejo de proyecto (minutas, planilla de riesgos, etc.)	20	Solo se incorporo las minutas.
Buscar herramientas en java para representar grafos en un browser.	30	Por ahora utilizaríamos JGraph

Historia de revisión

Actuante	Fecha	Comentarios
Diego Costanzo	2 de Mayo de 2004	Autor de versión inicial.
Diego Costanzo	5 de Mayo de 2004	Cambios en formato
Manuel Rodriguez	6 de Mayo de 2004	Sugerencia de cambiosy aprobación.
Jorge Coll	6 de Mayo de 2004	Aprobación
Hector Cancel	6 de Mayo de 2004	La es entregada en la reunión.

Minuta de Reunión #2 - 6/05/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 6 de Mayo de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Manuel Rodriguez	Miembro del Grupo
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Comentar el informe de elección de un lenguaje XML para grafos
- Comentar la primera versión del cronograma general del proyecto.
- Tener unan visión más clara del modelo conceptual.
- Definición de tareas para la próxima reunión de avance.

Desarrollo de Temas:

Comentar el informe de elección de un lenguaje XML para grafos

Se comento el informe de “Representación de grafos usando XML.doc” que elaboramos y nos hizo algunas sugerencias, como incluir referencias a la bibliografía en el texto del informe para que sea mas fácil de leer. Agregar lo de que graph pad no tiene la funcionalidad de poner atributos. Mencionar la herramienta web que dibuja grafos en Java. Va a leer el documento para hacer las correcciones que crea conveniente y aprobar o rechazar la selección del lenguaje elegido.

También surge de esta lectura dos requerimientos:

- Ver la posibilidad de implementar la funcionalidad de conversión de GXL a Heide
- Implementar en C o C++ un modulo que lea archivos GXL y los levante a alguna forma de representación en memoria.

Comentar la primera versión del cronograma general de proyecto.

Se nos recomienda que la elaboración del informe final, se vaya haciendo al finalizar cada una de las cuatro etapas que hemos distinguido. Quedo de mirarlo con más detenimiento.

Tener unan visión más clara del modelo conceptual.

El modelo debe estar influenciado para la aplicación de la confiabilidad diámetro acotada en redes P2P, por lo tanto trabajaremos con:

- G completo
- K = V
- TTL = D
- La confiabilidad de las aristas es la probabilidad de que esa arista este funcionando, valor que será expresado a través de la cantidad de conexiones promedio por nodo.
- La confiabilidad diámetro acotada. Probabilidad de que encontrar la información en no más de TTL pasos.
- El costo se mide por la cantidad de mensajes transmitidos, no puede haber mas de x cantidad de mensajes/minuto circulando por la red.
- Número de solicitudes de información promedio por minuto por nodo.

Se identifican en este proyecto dos problemas: La evaluación de la confiabilidad y la optimización de la probabilidad de que los nodos se comuniquen.

Problema de Evaluación

Entrada:
- Número de nodos del grafo
- TTL = D
- Cantidad de conexiones promedio por nodo.
- Cantidad de mensajes/minuto circulando por la red.
- Número de solicitudes de información promedio por minuto por nodo.

Salida: se busca encontrar la confiabilidad diámetro acotada.

Problema de Optimización

Entrada:
- Número de nodos del grafo
- Cantidad de mensajes/minuto circulando por la red.
- Número de solicitudes de información promedio por minuto por nodo.

Salida: se busca encontrar TTL y número de conexiones promedio por nodo

Tareas en curso:

Tarea	Status (%)	Comentarios
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	90	Faltan comentarios del profesor para realizar correcciones finales.
Definir y validar la estructura interna del grafo.	0	Definición y posterior aprobación del tutor
Definir un cronograma tentativo	70	Modificaciones menores, posterior aprobación del Tutor (mejorar la representación grafica)
Continuar leyendo trabajos relacionados al área.	60	Nos falta comentar el de Diameter Constrained Spanning Tree.
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	20	Solo se incorporo las minutas.
Buscar herramientas en java para representar grafos en un browser.	90	Contamos con la librerías Jgraph, y GXL y con los editores GXL graphpad y Jgraph pad. En principio con estas herramientas sería suficiente.
Applet para leer y visualizar grafos en formato GXL.	10	Se han estudiado las librerías y herramientas que usaremos.
Lectura de GXL en C/C++	0	Buscar librerías que faciliten el procesamiento de XML en estos lenguajes e implementar un modulo que las utilice.
Conversión GXL a Heidi en ambos sentidos.	40	De Heidi a GXL esta bastante avanzado

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	
Repasar métodos de simulación (Monte Carlo y otros)	0	

Historia de revisions:

Actuante	Fecha	Comentarios
Diego Costanzo	16 de Mayo de 2004	Autor de la versión inicial.
Manuel Rodríguez	17 de Mayo de 2004	Cambios y aprobación
Jorge Coll	17 de Mayo de 2004	Cambios y aprobación
Hector Cancela	17 de Mayo de 2004	Le es enviada por mail.

Minuta de Reunión #3 - 20/05/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 20 de Mayo de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Manuel Rodriguez	Miembro del Grupo
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Revisión de los documentos presentados la reunión anterior.
- Repaso del modelo para las etapas de optimización y evaluación.
- Avances del traductor HEIDI-GXL y desarrollo futuro.
- Avances en el visualizador grafos y desarrollo futuro.
- Bibliotecas para el manejo de XML en C/C++.
- Definición de tareas para la próxima reunión de avance.

Desarrollo de Temas:

Revisión de los documentos presentados la reunión anterior.

Se comentaron algunas correcciones en el informe de Representación de grafos usando XML entregado en la reunión anterior. Entre las mismas se destacan:

- El agregado de referencias a lo largo del mismo.
- Adecuar los objetivos
- Rever si es cierto que GXL no fue pensado como lenguaje de representación de grafos.
- Lo de la separación de la representación visual

A raíz del tema de las referencias en el documento, se aprovechó a definir una estructura para las mismas en los documentos siguientes del proyecto.
Dicha estructura consta de la concatenación de las iniciales de los autores (o las primeras letras del apellido en caso de tratarse de un solo autor) del documento que se referencia seguidas del año de publicación del mismo, separados por una coma. Por ejemplo [PK,01] para el documento de John Punin y Mukkai Krishnamoorthy de Junio del2001. En los casos especiales en los que el documento de referencia no tenga definido claramente los autores, se podrá utilizar una abreviación de su lugar de origen (por ej., en los casos de páginas Web) del mismo.
También se acordó presentar todas las referencias al final de los documentos y ordenadas en alfabéticamente.

Como una extensión del documento Representación de grafos usando XML, se planteó la realización de otro documento que ahonde más en el lenguaje elegido para la representación de grafos a lo largo del proyecto (GXL).
Algunos puntos a tratar en este documento serían:

- Extensión en la sintaxis de GXL (tomando como guía el documento Lenguajes canónicos para la descripción de Grafos de Pablo Rodríguez).
- Detalle de herramientas disponibles para el manejo de GXL.
- Discusión sobre las versiones existentes (1.0 y 1.1) y sus diferencias.
- Mencionar específicamente las fuentes de consultas existentes (mail-lists, newsgroups, etc.)

Tareas:

- Corregir el documento de Representación de grafos usando XML.
- Escritura de un documento más detallado sobre GXL.

Repaso del modelo para las etapas de optimización y evaluación.

A partir de la minuta de la reunión anterior, se plantean algunas correcciones al modelo planteado para la etapa de evaluación. El mismo quedaría de la siguiente manera:

Entradas:

- Número de nodos del grafo
- TTL = D
- Cantidad de conexiones promedio por nodo (confiabilidad de las aristas).
- Número de solicitudes de información promedio por minuto por nodo.

Salidas:
- La confiabilidad diámetro acotada
- Cantidad de mensajes/minuto generados en la red.

A propósito de este tema, se planteó también la forma en que se medirá (o estimará) la cantidad de mensajes circulando en el grafo, que será igual a la cantidad de envíos de mensajes de todos los nodos del grafo en el instante en cuestión.

También se definió que el modelo trabaja con dos "instantes de tiempo": uno define la topología del grafo, al definir que subconjunto enlaces del grafo completo están funcionando, y otro en el que dada la topología calculada se simula el envío de los mensajes y se realizan los cálculos necesarios según el caso (evaluación de la confiabilidad u optimización)

Avances del traductor HEIDI-GXL y desarrollo futuro.

El traductor HEIDI-GXL se encuentra en un estado de avance bastante importante, contándose con una versión del mismo probablemente para la próxima reunión.

En este punto se manejó la posibilidad de que el traductor inverso (de GXL a HEIDI) permita elegir tres atributos del grafo GXL y mapearlos a los tres atributos definidos en HEIDI.

Avances en el visualizador grafos y desarrollo futuro.

El otro desarrollo en curso es el de una aplicación para la visualización de grafos vía Web. La misma está en desarrollo, trabajándose actualmente en la reutilización de parte del código de una aplicación desarrollada en un Proyecto de Ingeniería de Software.

Bibliotecas para el manejo de XML en C/C++.

Dada la existencia de una gama considerable de bibliotecas, se manejó la posibilidad de realizar un análisis comparativo de las mismas y de empezar a ver que lenguaje se utilizará para los desarrollos en las etapas de optimización y evaluación de forma de buscar y profundizar en bibliotecas para ese lenguaje y entorno.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión de avance avanzar en las siguientes:

Tareas:

- Avanzar en los traductores HEIDI-GXL y GXL-HEIDI
- Avanzar en la aplicación para la visualización de grafos.

Tareas en curso:

Tarea	Status (%)	Comentarios
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	80	Se realizarán correcciones según sugerencias y se anexara informe específico sobre GXL.
Definir un cronograma tentativo	70	Modificaciones menores, posterior aprobación del Tutor (mejorar la representación grafica)
Continuar leyendo trabajos relacionados al área.	60	Nos falta comentar el de Diameter Constrained Spanning Tree.
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	50	Solo se incorporo las minutas. No se planea agregar nada más.
Buscar herramientas en java para representar grafos en un browser.	90	Contamos con la librerías Jgraph, y GXL y con los editores GXL graphpad y Jgraph pad. En principio con estas herramientas sería suficiente.
Applet para leer y visualizar grafos en formato GXL.	10	Se han estudiado las librerías y herramientas que usaremos.
Lectura de GXL en C/C++	0	Se intentará hacer un estudio comparativo de las herramientas disponibles.
Conversión GXL a Heidi en ambos sentidos.	40	De Heidi a GXL esta bastante avanzado; para GXL a HEIDI se definió posible forma de realizar la conversión.
Análisis de GXL como herramienta de representación de grafos.	0	Se hará un estudio más detallado de GXL, como una extensión del análisis comparativo de lenguajes para representación de grafos.

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	
Repasar métodos de simulación (Monte Carlo – Antitético Generalizado y otros de reducción de varianza)	0	

Tareas terminadas:

Tarea	Comentarios
Definir y validar la estructura interna del grafo.	Se definen los atributos de grafo, nodos y aristas. El profesor lo valido.

Historia de revisiones

Actuante	Fecha	Comentarios
Manuel Rodriguez	26 de mayo de 2004	Autor de la versión inicial.
Diego Costanzo	1º de Junio de 2004	Agrega comentarios y la aprueba.
Jorge Coll	3 de Junio de 2004	Agrega comentarios y la aprueba
Hector Cancela	3 de Junio de 2004	Le es enviada por mail y entregada en la reunión.

Minuta de Reunión #4 - 03/06/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 03 de Junio de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Manuel Rodríguez	Miembro del Grupo
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Comentar el avance obtenido en las semanas previas
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Comentar el avance obtenido en las semanas previas

Se comento que el avance obtenido en el visualizador de grafos, en las semanas anteriores no fue muy apreciable, debido a que parecíamos estar cerca de hacerlo andar pero todavía no veíamos resultados.

Se habló de que el traductor de Heidi a GXL esta casi terminado que estábamos comenzando a implementar el traductor de GXL a Heidi.

Tareas:

- Seguir avanzando con los visualizadores y los traductores.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión de avance avanzar en las siguientes:
Tareas:

- Avanzar en los traductores HEIDI-GXL y GXL-HEIDI
- Avanzar en la aplicación para la visualización de grafos.

Tareas en curso:

Tarea	Status (%)	Comentarios
Definir un cronograma tentativo	80	Modificaciones menores, posterior aprobación del Tutor (mejorar la representación grafica)
Continuar leyendo trabajos relacionados al área.	60	Nos falta comentar el de Diameter Constrained Spanning Tree.
Aplicación para la visualización de grafos representados en lenguaje GXL	50	Se está intentando reutilizar código de un Proy. de Ing. de Softw.
Applet para leer y visualizar grafos en formato GXL.	0	Se intentará realizar una reunión con un profesor de Sistemas de Información para evaluar alternativas de implementación.
Lectura de GXL en C/C++	0	Se intentará hacer un estudio comparativo de las herramientas disponibles.
Conversión GXL a Heidi en ambos sentidos.	50	De Heidi a GXL estaría terminado; para GXL a HEIDI se tiene definida la estructura de la aplicación pero no se comenzado a implementar.

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	
Repasar métodos de simulación (Monte Carlo – Antitético Generalizado y otros de reducción de varianza)	0	

Tareas terminadas:

Tarea	Comentarios
Definir y validar la estructura interna del grafo.	Se definieron los atributos de grafo, nodos y aristas.
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	Se incorporo la utilización de minutas para las reuniones con el tutor.
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	Se entregó un informe comparando diferentes lenguajes y se definió la utilización del lenguaje GXL.
Buscar herramientas en java para representar grafos.	Se utilizan la librerías Jgraph y GXL y los editores GXL graphpad y Jgraph pad.
Análisis de GXL como herramienta de representación de grafos.	Se entregó un informe con una descripción mas detallada de GXL.

Historia de revisiones

Actuante	Fecha	Comentarios
Jorge Coll y Manuel Rodriguez	17 de Junio de 2004	Autores de la versión inicial.
Diego Costanzo	17 de Junio de 2004	Aprobación
Hector Cancela	17 de Junio de 2004	Le es entregada
Hector Cancela	23 de Junio de 2004	Le es enviada por mail.

Minuta de Reunión #5 - 17/06/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 17 de junio de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodriguez	Miembro del Grupo

Agenda:

- Mostrar el avance en el visualizador de grafos.
- Mostrar el avance en el traductor Heide a GXL. y GXL a Heide
- Especificar el comportamiento del traductor GXL a Heidi.
- Definir tipo de documentación
- Definición Arquitectura de la solución
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar el avance en el visualizador de grafos.

Se instalo el J2 SDK 1.4.1 en la cual se ejecuto el visualizador de grafos y los traductores. Se estuvo un buen rato mirando como funcionaba, para identificar mejoras las cuales se detallan en la lista de tareas para la próxima reunión de avance.

Mostrar el avance en el traductores Heide a GXL y GXL a Heide

Se ejecutaron ambos traductores para grafo en particular. Se partió de un archivo GXL, el cual fue convertido a Heidi y luego con el traductor Heide a GXL, se lo hiz en el sentido inverso. Ese grafo así obtenido fue el que se uso para visualizar. Era un grafo puente con 4 nodos y 5 aristas. Ambas corridas fueron exitosas si bien falta en el traductor GXL implementar la interfase gráfica, la selección de los atributos e implementar la especificación que se detalla en el siguiente punto.

Especificar comportamiento traductor GXL a Heidi

Este conversor deberá proveer la funcionalidad de mostrar para cada atributo, de nodo y de arista la posibilidad de elegirlo de entre todos los atributos que aparecen para algún nodo del GXL y también los que aparecen a nivel del Grafo, diferenciando si es de Grafo o de nodo y dando la posibilidad que si eligió de Grafo pueda optar o no por sobrescribir con este valor el atributo en cuestión en caso que el nodo también lo tenga definido.

Definir tipo de documentación

La documentación será básicamente un diagrama de clases, una documentación de con formato de API de Java y algún diagrama UML más que se crea relevante. También para el caso de algoritmos complejos se deberá comentarlos en forma especial

Definición de Arquitectura de la solución

Se definió que tanto el visualizador como los traductores, trabaran sobre un subset de archivos GXL válidos. Se vera la posibilidad de implementar una función que dado un GXL valido diga si ese archivo es válido para nuestra aplicación. Un archivo gxl es esta bien formado para nosotros si define solo un grafo, los elementos del grafo solo son del tipo atributo, nodo o arista. Todos los atributos son simples y los únicos elementos de un nodo o arita son atributos.

Los traductores estarán integrados y serán una aplicación separada del visualizador. Será una aplicación y no un applet. Por otro lado el visualizador puede que sea una aplicación o un applet. Por mas detalle ver la primer tarea definida para la próxima reunión.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Definir si el visualizador será un applet o una aplicación que el usuario baje previamente, pero que sea asociable al Browser (Mozila, Netscape, Internet Explorer). Como cuando uno abre un pdf que esta en la web. Para esta decisión es importante evaluar cuanto demora en bajar el applet en una conexión dial-up y la complejidad de convertir la aplicación en applet.

- Desarrollar la interfase grafica para el visualizador GXL a Heide e implementar que sea seleccionable los atributos de confiabilidad, costo y capacidad para aristas y nodos.

- Integrar los dos traductores en una sola aplicación

- En el visualizador implementar los siguientes cambios:
    a) Que la arista no “se corte” por la existencia del rombo del cual cuelgan los atributos de la arista, que sea una sola línea recta.

    b) Incluir información de coordenadas *x* e *y* en el grafo, con un atributo booleano en el grafo que indique si tiene información grafica o no. Esta información será utilizada para mostrar en grafo con estos valores sustituyendo el actual algoritmo de minimización de cortes de aristas, para los grafos que tengan *x* e *y* definidos.

    c) Que se puede elegir o no ver atributos de arista y en forma separada ver o no información de atributos de nodos.

Tareas en curso:

Tarea	Status (%)	Comentarios
Applet o Aplicación para leer y visualizar grafos en formato GXL.	70	Faltan hacer las mejoras al visualizador identificadas en esta acta en la sección "Tareas para próxima reunión"
Traductores Heidi a GXL y GXL a Heidi	70	Falta implementar las tareas señaladas en esta acta de reunión en la sección "Tareas para la próxima reunión".
Documentar	0	Diagramas UML relevantes y JavaDoc
Función que valide Grafos Simples	0	Evaluar de hacerla vía DTD o en forma particular.
Definir de cronograma del proyecto.	80	Modificaciones menores, posterior es aprobación del Tutor (mejorar la representación grafica)
Continuar leyendo trabajos relacionados al área.	60	Nos falta comentar el de Diameter Constrained Spanning Tree.

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	Se estudiarán estas técnicas al comienzo de la fase 3 del proyecto.
Estudiar métodos de simulación (Monte Carlo – Antitético Generalizado y otros de reducción de varianza)	0	Se repasarán y/o estudiarán estos métodos al comienzo de la fase 2 del proyecto.
Lectura de GXL en C/C++	0	Se hará un estudio comparativo de las herramientas disponibles en la Fase 2 del proyecto.

Tareas terminadas:

Tarea	Comentarios
Definir y validar la estructura interna del grafo	Se definieron los atributos de grafo, nodos y aristas y fue validado por el tutor.
Análisis de GXL como herramienta de representación de grafos.	Se hizo informa detallado para GXL y fue validado por el tutor con fecha 7 de Junio por mail.
Buscar herramientas en java para representar grafos en un browser.	Se utilizan las librerías JGraph y GXL y los editores GXL Graphpad y JGraphPad . Ademas se reutilizó código de un proyecto anterior realizado por Manuel Rodriguez.
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	Se incorporó la utilización de minutas para las reuniones con el tutor.
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	Se realizaron correcciones y fueron aprobadas por el tutor con fecha 7 de Junio por mail.

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	19 de Junio de 2004	Autor de la versión inicial.
Manuel Rodríguez	21 de Junio de 2004	Sugerencias de cambio y aprobación
Jorge Coll	22 de Junio de 2004	Aprobación
Diego Costanzo	23 de Junio de 2004	Realización de Cambios
Hector Cancela	23 de Junio de 2004	Le es enviada por mail.
Diego Costanzo	8 de Julio de 2004	Se agregan tareas.

Minuta de Reunión #6 - 8/07/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 08 de julio de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Mostrar el avance en el visualizador de grafos.
- Mostrar el avance en el traductor Heidi a GXL y GXL a Heidi
- Comienzo de la fase 2 “Evaluación de la Confiabilidad Diámetro Acotada”
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar el avance en el visualizador de grafos

Se ejecuta la última versión del visualizador, con dos o tres grafos constatando la mejoras en la aplicación desde la versión anterior. En particular se muestra que los atributos son desplegados con su correspondiente valor, a su vez los de las aristas “cuelgan” del punto medio de esta en el lugar del rombo de la versión anterior, la arista no se corta en la primer visualización y están implementadas las funcionalidades de mostrar o no los atributos de los nodos y de las arista.
Se explicitan las mejoras pendientes que se detallan más abajo en la definición de tareas para la próxima reunión de avance.

También nos pregunto si habíamos optado por rectángulos para los nodos por alguna razón en particular porque de no afectar demasiado sería mejor sustituirlos por círculos que son mas naturales.

Mostrar el avance en el Traductor Heide a GXL y GXL a Heide

Se ejecuta la última versión del traductor, con dos grafos verificando las mejoras en la aplicación desde la versión anterior. En particular se muestra que ambos traductores ahora están consolidados en una sola aplicación, el traductor GXL a Heidi, permite ahora seleccionar los archivos de origen y destino y lo hacen a través de un botón examinar y además permite mapear que atributos del archivo GXL a nivel de Grafo o de nodo y arista serán mapeados a la confiabilidad, costo y capacidad del archivo Heidi.

Uno de los dos archivos GXL no tenia id en las aristas, lo que provocó al utilizar el archivo generado para traducir desde Heidi a GXL, la aplicación diera error de formato en archivo Heidi. Esto se debe a que en Heidi no esta permitido id iguales y estaban todos en *null*. Se agregó este error (general un línea en blanco en vez de null para aristas sin id) a la lista de correcciones a realizar y esto motivo la lectura de las tareas pendientes en esta aplicación, las cuales están documentadas al final de esta acta. En particular para la de "best practices" en Interfases de graficas nos cito un libro al respecto pero cuya aplicación era mas bien para aplicaciones de mayor porte. Con respecto a si partir en 3 pasos el wizard en la conversión de GXL a Heidi nos dijo que estaba bien, como esta actualmente.

Comienzo de la fase 2 "Evaluación de la Confiabilidad Diámetro Acotada"

Se le propone al tutor, comenzar con la siguiente fase del proyecto a pesar de no haber terminado con la anterior para no retrasar el cronograma originalmente trazado e ir terminando con el visualizar y el traductor en el transcurso de las fases 2 y 3. Esta opción le parece bien y de esta manera se comienza la fase de evaluación de la confiabilidad diámetro acotada, dándonos bibliografía del método de Monte Carlo y explicándonos algunas conceptos para implantar este algoritmo así como el que lo resuelve por un método exacto. Estas ideas las tratamos de plasmar en los siguientes párrafos, como manera de clarificarlas y validarlas.

Se define una función Φ que dada una posible instancia de aristas del grafo devuelve un valor según un determinado criterio. Por ejemplo puede devolver 0 o 1 según para todos los nodos del grafo haya un camino de largo menor o igual que D, o un valor entre 0 y 1 indicando el porcentaje de pares de nodos que están a distancia menor o igual que D.

Luego el método exacto consiste en evaluar la Φ para todos los posibles estados del grafo. Se va sumando los valores obtenidos en cada iteración ponderado por la probabilidad de ocurrencia de ese grafo.

El método Monte Carlo vainilla es básicamente con la misma idea pero en vez de elegir todos los posibles grafos se toma una muestra aleatoria *m*. En cada una de las *m* iteraciones, para cada arista se sortea un numero entre 0 y 1 si este numero es menor al cociente CanConPromPorNodo/(N-1), la elijo en caso contrario no. A esa instancia así obtenida le aplico la función Φ y acumulo el resultado. Al final del método se devuelve el valor acumulado/m.

Trabajaremos con 3 funciones Φs. Una para determinar la confiabilidad diámetro acotada, otra para determinar el porcentaje de pares de nodos que su distancia es menor o igual que D y la tercera para determinar la carga de la red. La primera para todo par de nodos se fija si el camino más corto es menor que D, si es así devuelve 1 y si no 0. La segunda para todo par de nodos se fija si el camino mas corto es menor que D, solo si es así cuenta este par. Luego devuelve la fracción de pares que cumplen esa condición. La tercer Φ mide la cantidad de paquetes transmitidos en la red, sumamos la cantidad de paquetes transmitidos por la inundación de cada nodo, en principio asumiendo nodos "sin inteligencia" y multiplicamos por la cantidad de pedidos por minuto se estima realice cada nodo.

Todo esto se puede hacer para el problema Rv o Rsk. El problema Rv es cuando interesa calcular la confiabilidad todo el conjunto de nodos, mientras que en el problema Rsk, se toma en cuenta la confiabilidad desde un nodo fuente a un set de nodos terminales *k*.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Implementación de método exacto en Java y C/C++
- Seudocódigo de Monte Carlo vainilla.

- Avances en Visualizador
  - Bug de doble quiebre en arista
  - Círculos en vez rectángulos
  - Redibujar arista cuando se mueve un nodo
  - Implementar coordenadas X e Y en formato de GXL para visualización.
  - Correcciones y mejoras varias (ver documentación interna)
- Avances en Traductores
  - Testeo exhaustivo (ciclo GXL a Heidi y viceversa)
  - Validar secuencia de eventos
  - Incorporar el Botón Anterior, para ir hacia atrás en el wizard.
  - Buscar e implementar algún documento de Best Practice para Interfaces gráficas
  - Correcciones y mejoras varias (ver documentación interna)
- Documentar ambas aplicaciones
- Hacer un validar de Grafo Simple que lo usen ambas aplicaciones (DTD)

Tareas en curso:

Tarea	Status (%)	Comentarios
Applet o Aplicación para leer y visualizar grafos en formato GXL.	85	Faltan hacer las mejoras al visualizador identificadas en esta acta en la sección "Tareas para próxima reunión"
Traductores Heidi a GXL y GXL a Heidi	85	Falta implementar las tareas señaladas en esta acta de reunión en la sección "Tareas para la próxima reunión".
Implementar el método exacto en Java y C/C++	0	
Llevar un Seudocódigo del método Monte Carlo vainilla	0	
Documentar Visualizador y Traductor	0	Diagramas UML relevantes y JavaDoc
Función que valide Grafos Simples	0	Evaluar de hacerla vía DTD o en forma particular.
Estudiar métodos de simulación (Monte Carlo – Antitético Generalizado y otros de reducción de varianza)	10	Se leyo el método Monte Carlo vainilla.
Definir de cronograma del proyecto.	80	Modificaciones menores, posterior es aprobación del Tutor y mejorar la representación grafica.
Continuar leyendo trabajos relacionados al área.	50	Nos falta comentar el de Diameter Constrained Spanning Tree y leer el de Monte Carlo especifico para confiabilidad.

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	Se estudiarán estas técnicas al comienzo de la fase 3 del proyecto.
Lectura de archivos GXL en C/C++	0	Se hará un estudio comparativo de las herramientas disponibles en la fase 2 del proyecto.

Tareas terminadas:

Tarea	Comentarios
Definir y validar la estructura interna del grafo	Se definieron los atributos de grafo, nodos y aristas y fue validado por el tutor.
Análisis de GXL como herramienta de representación de grafos.	Se hizo informa detallado para GXL y fue validado por el tutor con fecha 7 de Junio por mail.
Buscar herramientas en java para representar grafos en un browser.	Se utilizan las librerías JGraph y GXL y los editores GXL GraphPad y JGraphPad . Además se reutilizó código de un proyecto anterior realizado por Manuel Rodriguez.
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	Se incorporó la utilización de minutas para las reuniones con el tutor.
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	Se realizaron correcciones en el informe y fueron aprobadas por el tutor con fecha 7 de Junio por mail.

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	10 de Julio de 2004	Autor de la versión inicial.
Todo el grupo	12 de Julio de 2004	Lectura, modificaciones y aprobación.
Hector Cancela	14 de Julio de 2004	Le es enviada por mail

Minuta de Reunión #7 - 29/07/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 29 de julio de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodriguez	Miembro del Grupo

Agenda:

- Mostrar el avance en el visualizador de grafos.
- Mostrar el avance en la aplicación de Evaluación
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar el avance en el visualizador de grafos

Se ejecuta la última versión del visualizador, para determinar la forma de representar gráficamente los nodos. Se eligió que fueran cuadrados en vez de círculos, dado que por una propiedad de Jgraph si se elige los círculos las aristas son dibujadas, hasta donde estaría el cuadrado y no quedaba bien visualmente.

Además se mostró el arreglo hecho en la selección de más de un elemento. En efecto si ahora se elige más de un elemento todos son redibujados.

Mostrar el avance en la aplicación de Evaluación

Se mostró la primer versión de esta aplicación donde están implementados en Java, las tres funciones Φ's, para los métodos exactos y Monte Carlo Standar.

Se ve la necesidad que se despliegue la varianza y eventualmente el intervalo +/- 3 sigma, para el Método Monte Carlo, así como el resultado de la formula de aproximación para la Φ de carga.

Analizamos que C++, sería más apropiado que C, pues sería más rápida la transformación del código Java en ese lenguaje, logrando una performance semejante si se trabaja con un estilo de programación que apunte en ese sentido. También nos sugiere que en la versión de C++,

reservemos toda la memoria al principio y la liberemos al final, este factor es importante para optimizar el tiempo.

También comentamos que una diferencia a implementar en C++, es la de pasar como parámetro un array con las 3 funciones a cada método para computar las tres funciones en la misma iteración.

Comentamos nuestra idea de invocar a las rutinas de C++, desde la aplicación Java mediante JNI, para mantener un ambiente unificado y facilitar la utilización de la aplicación.

Hablamos de realizar una prueba de validación que compare ambos métodos implementados hasta el momento para grafos pequeños y para los casos en que solo el Monte Carlo es posible correrlo, validar los resultados de este probándolo contra si mismo, al variar solo uno de los parámetros y observando si la salida cambia como se espera. También se puede incluir graficas como por ejemplo variación de la Confiabilidad versus la D.

Otras modificaciones detectadas fuero la de guardar los pedidos por unidad de tiempo como un flota en vez de entero, disminuir la precisión al desplegar los resultados y averiguar si el tiempo que se muestra al final es tiempo real o de CPU.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, detener por un momento la implementación de nuevos algoritmos para empezar a redondear todo lo avanzado hasta el momento. En ese sentido se identificaron las siguientes tareas, aunque no todas ellas van a poder atenderse para la próxima reunión de avance.

Tareas:
Aplicación Visualizador:

- Implementar X,Y y salvar.
- El scroll
- Que no aparezcan fuera de la pantalla
- Que tome el look and fell de Windows (ambiente)
- El refresh inicial
- Bugs del doble quiebre de la arista y cuando quedan varios nodos cercanos.
- Hacer que sea web.
- Probar el poner atributos con el mismo nombre en los nodos rel y grafo
- Correcciones y mejoras varias (ver documentación interna)

Aplicación Evaluación

- Que muestre la varianza (y tal vez el rango +/- 3 sigma)
- Que muestre la estimación de la cuenta mal para la carga
- Informe de análisis de resultado uno vs el otro el Monte Carlo consigo mismo.
- Algoritmos en C++, pasando como función para evaluar las 3 Φ's al mismo tiempo.
- Invocar a C++ desde Java con JNI.
- Optimizar todos los algoritmos
    - Que la matriz sea array de booleanos
    - Optimizar BFS de forma que pare cuando llega a una profundidad mayor que D
    - Ver otras
- Mejoras en la interfaz
    - Diseñar interfaz final
    - Validación de parámetros
- Correcciones varias
    - Casos de borde (analizar 0 y 1 para los 4 parámetros).
    - Numero de pedidos por Nodo como double
    - Menor precisión al desplegar resultados

Aplicación Traductores
- Agregar Botón Anterior, para ira hacia atrás en el wizard
- Validar secuencia de eventos
- Correcciones y mejoras varias (ver documentación interna)

Común a las 3 aplicaciones:
- Hacer un setup
- Implementar una UI según Best Practise
- Javadoc
- Diagrama de clases
- Diagrama de componentes
- DTD para validar que sea un grafo simple (no aplica para Aplica Evaluación)
- Manual de usuario
- Testing exhaustivo

Administrativo:
- Modificar el project

Mejoras:
- Mejorar el algoritmo de ordenación
- Probar el poner atributos con el mismo nombre en los nodos rel y grafo

Tareas en curso:

Tarea	Status (%)	Comentarios
Aplicación Evaluación	60	Resta implementar el Algoritmo Monte Carlo Antitético y las mejoras señaladas en esta acta en la sección “Tareas para próxima reunión”
Applet o Aplicación para leer y visualizar grafos en formato GXL.	85	Faltan hacer las mejoras al visualizador identificadas en esta acta en la sección “Tareas para próxima reunión”
Traductores Heidi a GXL y GXL a Heidi	85	Falta implementar las tareas señaladas en esta acta de reunión en la sección “Tareas para la próxima reunión”.

Tareas próximas a realizar:

Tarea	Status (%)	Comentarios
Estudiar meta heurísticas (ej Algoritmos Genéricos, Tabú Search y otras)	0	Se estudiarán estas técnicas al comienzo de la fase 3 del proyecto.
Lectura de archivos GXL en C/C++	0	Se hará un estudio comparativo de las herramientas disponibles en la fase 2 del proyecto.

Tareas terminadas:

Tarea	Comentarios
Definir y validar la estructura interna del grafo	Se definieron los atributos de grafo, nodos y aristas y fue validado por el tutor.
Análisis de GXL como herramienta de representación de grafos.	Se hizo informa detallado para GXL y fue validado por el tutor con fecha 7 de Junio por mail.
Buscar herramientas en java para representar grafos en un browser.	Se utilizan las librerías JGraph y GXL y los editores GXL GraphPad y JGraphPad . Además se reutilizó código de un proyecto anterior realizado por Manuel Rodriguez.
Implementar el método exacto en Java y C/C++	
Continuar leyendo trabajos relacionados al área.	
Evaluar alguna técnica para la gestión de proyecto (minutas, planilla de riesgos, etc.)	Se incorporó la utilización de minutas para las reuniones con el tutor.
Evaluar y seleccionar un lenguaje basado en XML para la representación de grafos.	Se realizaron correcciones en el informe y fueron aprobadas por el tutor con fecha 7 de Junio por mail.

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	5 de Agosto de 2004	Autor de la versión inicial.
Manuel Rodriguez	6 de Agosto de 2004	Correcciones y Aprobación
Jorge Coll	6 de agosto de 2004	Aprobación
Hector Cancela	8 de agosto de 2004	Enviada a Cancela por correo electrónico

Minuta de Reunión #8 - 26/08/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 26 de Agosto de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodriguez	Miembro del Grupo

Agenda:

- Mostrar el avance en Visualizador y aplicación de Evaluación
- Mostrar y distinguir mejoras de la prueba de validación
- Hablar acerca de la evaluación de mitad de proyecto
- Consultas varias
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar el avance en Visualizador y aplicación de Evaluación

Por razones de tiempo no se ejecutaron ninguna de las dos aplicaciones, pero se mencionaron los avances hechos desde la reunión anterior relacionados a la misma, en particular se menciono:

En lo que respecta al Visualizador de grafos:

- El visualizador permite salvar el archivo guardando las coordenadas donde se tenía ubicado cada nodo y al abrirlo lee dicha información. No se almacena la información de posición de los atributos.
- También toma el look and feel del ambiente donde esta corriendo, en vez de usar el default como hasta la versión anterior.

Mientras que para la aplicación de evaluación se avanzó en lo siguiente:

- Ahora tiene una única clase *Monte Carlo Crudo* y una única clase *Generación Completa de Estados*, a las que se le pasa como parámetro un array de objetos de clase Fi. Esto permite evaluar más de una misma Fi, en la misma ejecución. La clase Fi es abstracta y declara el método *evaluar*. Los métodos también son clases que derivan de la clase abstracta *Método* que declara el método *calcular*.
- Se arregló un bug que presentaban la Fi de Diámetro Acotada y la Fi de Porcentaje el cual sobrescribía valores de distancias ya calculadas anteriormente.
- Se documento la API de la aplicación con la funcionalidad de JavaDoc
- Se hizo en papel el diagrama de clases y el diagrama de colaboración.

- Se empezó a analizar como sería el Monte Carlo Antitético.
- Se implementaron las 3 Fi's y los dos métodos en C++ (en principio, no se implementará el antitético en C++). La conexión JNI, si bien esta hecha hay un detalle por el cual no funciona.
- Se hicieron mejoras menores detectadas en la reunión anterior como:
    - Se muestra el sigma, el rango y la estimación de la carga
    - Permite elegir más de un lenguaje y más de una Fi a la vez.
    - Se cambio el formato del tiempo y se puede elegir la precisión del resultado.

Mostrar y distinguir mejoras en la prueba de validación

Se mostraron los resultados de la prueba de validación, la cual en líneas generales le pareció bien. Se sugirió incluir otras dos validaciones:

- Una analítica, hallando la confiabilidad diámetro acotada analíticamente para casos de 3 y 4 nodos y otra
- Contra un resultado conocido de confiabilidad clásica (D=N-1), para lo cual nos envió un paper con varios resultados.

Se observo que la forma como se estaba calculando la varianza no era la correcta.

Se distinguieron, además, algunas mejoras para las pruebas de validación:

- Tomar 3 cifras significativas en lugar de 3 lugares después de la coma para los resultados decimales.
- Usar escalas logarítmicas en las gráficas.
- Poner la varianza en los resultados de la prueba de MCC y mostrar varios resultados del caso tomado como base en el MCC, para constatar que el 95 % de los veces el resultado cae en el rango +/- 3 sigma centrado en el valor exacto.
- Buscar información más detallada sobre el generador de números aleatorios que brinda Java y buscar la forma de poder pasarle la semilla de generación, de forma de poder repetir los experimentos.
- También se comento la posibilidad de hacer una implementación más genérica de los algoritmos, pasando como parámetro una matriz de probabilidades, la cual en el caso todos sus valors serían el mismo (cant_conexiones/N-1).
- Rever el cálculo del sigma teórico que estamos utilizando, ya que el mismo es válido solo para la confiabilidad "común".

Hablar acerca de la evaluación de mitad de proyecto

Se decidió hacer un informe de avance del proyecto, para ser presentado en la evaluación de mitad de proyecto. También se evaluara hacer una presentación. Este informe sería una inversión de tiempo para el informe final y abarcaría 2 de las 3 fases del proyecto, la primera que es de representación y visualización de grafos basados en XML y la segunda que es la de la evaluación de la confiabilidad y otras medidas.

Para la próxima reunión estaríamos presentando el índice y un borrador del informe.

Consultas Varias

Se comentan las respuestas a las consultas más relevantes:

Para tamaño de muestra igual a 1, no estaría definida, este caso se lo puede no permitir.

Implementación en C++, si se confirman los resultados que en C++, los tiempos de ejecución son nada más que la mitad en Java, sería razonable elegir java e implementar los algoritmos de optimización en este lenguaje.

Acerca de la consolidación de las 4 aplicaciones se concluyo que no tendría mucho sentido, dado que lo que estamos desarrollo es un conjunto de herramientas.

Sobre el cálculo de Sigma en los algoritmos de Monte Carlo con cantidad de muestras 1, el mismo no se encuentra definido.

Definición de tareas para la próxima reunión de Avance

Para la siguiente reunión de avance se definió avanzar en las siguientes tareas:

- Hacer el índice y un borrador del informe de mitad del proyecto.
- Implementar el Antitético generalizado
- Mostrar resultados de la evaluación en una tabla
- Terminar la prueba de validación y hacer el informe de la misma
- Terminar la conexión de Java con C++ vía JNI

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	60	Básicamente resta implementar el Algoritmo Monte Carlo Antitético, mostrar el resultado en una tabla, integrarlo con la dll de C++ y terminar con el informe de validación.
Aplicación de Optimización	0	Aun no comenzada
Applet o Aplicación para leer y visualizar grafos en formato GXL.	85	Resta documentar e implementar detalles menores, ver planilla de “Tareas Pendientes v 1.4”
Traductores Heidi a GXL y GXL a Heidi	85	Resta documentar e implementar detalles menores, ver planilla de “Tareas Pendientes v 1.4”

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	28 de Agosto de 2004	Autor de la versión inicial.
Manuel Rodríguez	30 de Agosto de 2004	Se agregan algunos apuntes sacados durante la reunión.
Jorge Coll	1 Setiembre de 2004	Aprobación
Hector Cancela	2 Setiembre de 2004	Se le envía acta a Cancela

Minuta de Reunión #9 - 16/09/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 16 de Setiembre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodriguez	Miembro del Grupo

Agenda:

Desarrollo de Temas:

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación		
Aplicación de Optimización		
Applet o Aplicación para leer y visualizar grafos en formato GXL.		
Traductores Heidi a GXL y GXL a Heidi		

Historia de revisiones

Actuante	Fecha	Comentarios

Minuta de Reunión #10 - 23/09/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 23 de setiembre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo
Diego Costanzo	Miembro del Grupo

Agenda:

- Tener un retorno sobre la documentación de los traductores ya entregada
- Obtener lineamientos para poder comenzar con la fase de optimización
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Tener un retorno sobre la documentación sobre los traductores ya entregada

En el correr de la semana previa a la reunión se le envió al profesor un informe sobre el traductor Heidi-GXL, para que éste realizara las correcciones que crea pertinente, de forma que nos sirva como guía para analizar el resto de las aplicaciones que forman parte del proyecto.

Se nos realizo las siguientes observaciones sobre la documentación entregada para el traductor:

- Falta que se integre con el resto de la documentación, no debemos perder de vista que forma parte del informe global. En particular se señalo no tenia demasiado sentido que ésta sección del informe tuviese objetivos y conclusiones, sino que estos estarían dentro del capitulo en el cual la sección esta contenida.
- Se observo que para las referencias que son a más de un autor solo se utilice iniciales de los mismos y no abreviaciones como aparecía en el informe.
- Los diagramas de colaboración que forman parte del informe están hechos a un nivel de detalle muy alto que no es en un principio el necesario.

Obtener lineamientos para poder comenzar con la fase de optimización

El objetivo de la optimización es hallar los valores de D y C (Cantidad de Conexiones promedio por Nodo) de manera de optimizar la confiabilidad diámetro acotada, sujeto a la restricción de carga del la red (paquetes transmitidos en la red por unidad de tiempo). Las variables D y C generan un espacio de soluciones el cual puede verse como discreto, si C varia en los Naturales o continuo si C varia en los Reales.

A su vez podemos hablar de algoritmos para la optimización que son:

- Exhaustivos
- Analíticos
- Heurísticos

Se observó que la elección del algoritmo a utilizar en la fase de optimización dependerá fuertemente del tiempo que se demore en evaluar las distintas Fi`s. Por tal motivo se vio que para la próxima reunión sería interesante contar con esta información.

Otros

El tutor nos informo que en el mes de diciembre se ausentará del país, para que mitiguemos el impacto que esto pudiese causar sobre el objetivo de terminar el proyecto en diciembre. Durante el periodo en que éste este ausente, no tendremos las reuniones habituales, pero si seguiremos en contacto vía mail donde podremos sacarnos las dudas. Por tal motivo el tutor nos aconsejo que priorizáramos el informe para ya tenerlo avanzado antes de la fecha mencionada.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Probar para que valores de N y D es razonable (en términos de tiempo y memoria) aplicar el algoritmo de evaluación.
- Generalización del antitético
- Buscar error en el sigma obtenido en nuestra aplicación
- Avanzar con el informe

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	85	Básicamente resta, terminar informe de validación, ver bug del sigma y del antitético en red Montevideo.
Aplicación de Optimización	5	Se esta empezando a evaluar la estrategia a utilizar
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Traductores Heidi a GXL y GXL a Heide	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Jorge Coll	29 de Setiembre de 2004	Autor de la versión inicial.
Todo el grupo	30 de Setiembre de 2004	Lectura, modificaciones y aprobación.
Hector Cancela	30 de Setiembre de 2004	Le es enviada por mail

Minuta de Reunión #11 - 30/09/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 30 de setiembre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Mostrar avances desde la reunión anterior
- Lineamientos para algoritmos de Optimización
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar avances desde la reunión anterior

Se comento que se mejoró sensiblemente la performance en el cálculo de la confiabilidad diámetro acotada, al eliminar los nodos accedidos antes de ponerlos en la cola, en vez de después. Para el caso k10 por ejemplo, se paso de 100 segundos a 35.

También se mejora la performance en el calculo de la confiabilidad al no evaluar los casos en que las cantidad de aristas en menor a N-1. Con este cambio en la red de Montevideo, por ejemplo, de resolverse en 130 segundos pasá a 85.

Se informó que el sigma esta bien calculado, simplemente que en algunos casos da mal por un error numérico. Con algunos datos particulares da valores incorrectos, NaN o valores mayores a 1. En ese momento surge la idea de castear los enteros a doubles.

Se comentó que la observación de que en C++ los valores daban en general más altos que en Java es por un tema del generador del números aleatorios. De hecho se hizo la prueba de que en C++ calcule con los mismos valores aleatorios que en Java y dio lo mismo.

Se mostró la tabla para Analizar la pregunta de hasta que N es razonable aceptar para la optimización

Se comento que se Redefinió la clase GrafoEvaluación, ahora consta de una matriz de aristas que representa que aristas hay en el grafo y con que confiabilidad y otra matriz de booleanos que modela el estado de un grafo.

Lineamientos para algoritmos de Optimización

Se descartó trabajar sobre el problema de doble optimización mencionado en al reunión anterior y a su vez se decidió enfocarnos solo sobre un problema de los otros dos posibles, por ejemplo el de maximizar la confiabilidad, sujeto a la carga, aunque se tratara de hacer los algoritmos genéricos como para que la resolución del otro problema (minimizar la carga sujeto a una restricción de confiabilidad) fuera meramente intercambiar las evaluaciones de una y otra función. También se definió trabajar sobre un espacio discreto.

Se observó que la confiabilidad diámetro acotada, tanto en función del diámetro, como de la cantidad de conexiones es una función creciente. Lo mismo sucede con la carga, por lo tanto se llego a que un buen algoritmo para resolver el problema es el siguiente:
En el espacio de soluciones que es una matriz de N-1 filas por N-1 columnas (para fijar ideas supongamos que en las filas varia el C, en las columnas el D y que estamos en el problema de maximizar la confiabilidad) se va buscando (por búsqueda lineal o binaria) la celda a partir de la cual la carga no cumple la restricción
Por monotonía sabemos que el resto de la fila tampoco va a cumplirla y en general para todas las filas (con C mayores) sucederá lo mismo.
Por tanto se pasa para la fila de abajo y ahí se vuelve a buscar pero ahora entre 1 es esa columna por la que se bajo.
Finalmente queda definida una frontera, la cual hay que recorrer para saber en cual de esos puntos maximiza la confiabilidad.

A pesar de que este método es exacto, no obtendremos un valor tal dado que utilizaremos para evaluar la carga y la confiabilidad, un algoritmo basada en Monte Carlo. Para manejar la acumulación de errores, hay dos formas, comparando los intervalos de confianza (si se intersectan entonces ambos valores son igualmente buenos) o al que voy guardando como mejor lo vuelvo a comparar.

La monotonía de ambos funciones si bien es posible y se bosquejó alguna idea no es necesario demostrarla.

Se vio que era importante hacer un análisis del error de la formula de carga respecto al algoritmo para ver si había alguna forma de poder usar la formula, dado que los tiempos de calculo crece exponencialmente al crecer D.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Avanzar con el informe
- Escribir diseño y seudocódigo del algoritmo de optimización
- Analizar relación de formula de carga con algoritmo de carga

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	85	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	10	Se esta evaluando el algoritmo a implementar
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Traductores Heidi a GXL y GXL a Heide	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	11 de Octubre de 2004	Autor de la versión inicial.
Maule Rodriguez	12 de Octubre de 2004	Lectura, modificaciones y aprobación.
Hector Cancela	13 de Octubre de 2004	Lectura, modificaciones y aprobación.
Hector Cancela	14 de Octubre de 2004	Le es enviada por mail

Minuta de Reunión #12 - 14/10/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 14 de octubre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Mostrar Avances desde la reunión anterior
- Estudiar la posibilidad de utilizar una formula que calcule la carga de la red al implementar los algoritmos de Optimización
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Mostrar avances desde la reunión anterior

Se mostró el estudio que compara la carga real de la red vs. la carga calculada con la formula: N*Pe*C*((C-1)D -1)/(C-2), sobre el cual se estuvo trabajando casi toda la reunión de avance con el objetivo de sacar alguna conclusión para el diseño del algoritmo.

Se comento que quedó finalmente resuelto el error en el cálculo de sigma. Eran dos errores, el que descubrimos el la reunión anterior y la función de conversión de double a string para los casos que el resultado estaba en notación científica

El algoritmo va mostrando el porcentaje de avance, por ahora en la salida estándar.

También se señalo que se mejor sensiblemente el tiempo de ejecución de la función de cargo, básicamente por dos cambios:

- Se agregan menos elementos en la cola, al no ingresar los elementos que están a distancia d+1, con este cambio para algunos casos se mejor el tiempo en el orden de unas 10 veces
- La cola se implemento como un array circular lo que en algunos casos llevo a una mejora de más del 50% del tiempo.

Finalmente se comentaron las ideas sobre las que estuvimos trabajando, para el algoritmo de optimización, como el tema de la “frontera de unos”.

Estudiar la posibilidad de utilizar una formula que calcule la carga de la red al implementar los algoritmos de Optimización

Mirando el estudio de la carga realizado se llego a que habría que realizarle las siguientes modificaciones para poder sacar mejores conclusiones:

- Calcular el error cometido con respecto al valor real
- En la hoja que se varía la cantidad de nodos, hacerlo con el resto de los parámetros iguales a las hojas anteriores.
- Poner las varianzas de todos los cálculos.
- Poner unidades en los ejes de los gráficos.
- Correr todo el estudio de carga utilizando la formula resultante de que cada nodo manda el paquete a todos sus nodos adyacentes.
- Realizar el estudio de comportamiento de la carga al variar el D con tamaños de muestra mayores y menores a 50.
- Probar de correr los estudios anteriores con tamaños de muestra menores a los utilizados hasta el momento (1000) para que el error de la evaluación, aunque sea mayor que al calcularlo con más muestras, siga siendo mejor que la utilización de la formula. Como resultado debemos obtener más rapidez en los cálculos y menor precisión.

También se planteo que en el algoritmo de optimización el tamaño de las muestras que se utilice para evaluar el grafo no tiene por que ser fijo sino que puede depender del error que se esta cometiendo.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Terminar el estudio de carga y mandarlo dentro de los próximos 4 días.
- Avanzar con el informe

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	90	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	10	Se esta evaluando el algoritmo a implementar
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Traductores Heidi a GXL y GXL a Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Jorge Coll	21 de Octubre de 2004	Autor de la versión inicial.
Manuel Rodríguez	21 de Octubre de 2004	Aprobación
Diego Costanzo	21 de Octubre de 2004	Agregar algunos puntos y aprobación
Héctor Cancela	21 de Octubre de 2004	Le es enviada por mail.

Minuta de Reunión #13 - 21/10/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 21 de octubre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Analizar los resultados obtenidos con la fórmula “nueva” para la carga de la red al implementar los algoritmos de optimización.
- Definir el problema a resolver en la optimización y rasgos generales en la implementación del mismo.
- Definición de conceptos referentes a la Confiabilidad Diámetro Acotada.
- Apliaciones a entregar
- Definición de tareas para la próxima reunión de avance

Desarrollo de Temas:

Analizar los resultados obtenidos con la fórmula “nueva” para la carga en la red al implementar los algoritmos de optimización.

La fórmula para la carga que denominaremos “nueva” consiste en calcular N * Pe * C *(C^D - 1)/(C) y la misma es una variante de la fórmula original (que denominaremos fórmula “vieja”) cambiando C-1 por C.

Los resultados experimentales obtenidos indican un buen comportamiento de la misma, aparentemente sobreestimando los valores reales de la carga y con un porcentaje de error respecto a ésta que se disminuye o aumenta muy lentamente, dependiendo del caso que se trate (variando N, D o C)
Estos resultados son relativos, debido a que surgen de un experimento particular para un juego de TTL, pedidos por nodo y cantidad de conexiones promedio por nodo particular. De todas formas se decidió utilizar ésta fórmula para la optimización.

Definir el problema a resolver en la optimización y rasgos generales en la implementación del mismo.

Habiéndose definido el uso de la fórmula para la carga, se tomaron algunas decisiones con respecto al programa de optimización:

- Inicialmente el problema que se atacará será el de maximizar la Confiabilidad sujeto a una restricción en la carga. Se comenzará haciendo evaluaciones de la carga en el espacio de muestras para encontrar la frontera de factibilidad del problema, y luego se recorrerá ésta en busca de la máxima confiabilidad usando el algoritmo de evaluación para la misma.
- La evaluación de la carga se podrá hacer de varias formas:
    - Utilizando la fórmula “vieja”
    - Utilizando la fórmula “nueva”
    - Utilizando el algoritmo de evaluación de la carga

  La decisión de qué método usar se puede hacer a nivel del programa dependiendo del conjunto de parámetros ingresado, o puede llegar a ser una decisión del usuario de la aplicación. Este punto se decidirá más adelante.
  La posibilidad de repetir corridas del programa de optimización variando el método con el cual se evalúa la carga, permite realizar estudios comparativos interesantes.
- No tiene sentido trabajar el problema de minimizar la carga sujeto a una restricción de confiabilidad, ya que la utilización de una fórmula estimativa para la evaluación del objetivo generará resultados con un margen de error inaceptable.

En definitiva, la aplicación de optimización tendrá como entradas el N (cantidad de nodos) y la restricción de carga sobre la solución; mientras que las salidas serían la pareja (D, C) que maximiza la confiabilidad cumpliendo con la restricción ingresada, y eventualmente la posibilidad de obtener un archivo de texto con toda la frontera de soluciones evaluadas.

Definición de conceptos referentes a la Confiabilidad Diámetro Acotada.

Ante una consulta particular surgió un repaso sobre algunos conceptos referentes a la Confiabilidad Diámetro Acotada, los cuales para su definición repasamos a continuación:

- Una confiabilidad R_V indica:
    - el porcentaje de tiempo en que la red no induce pérdida de información; o análogamente
    - el porcentaje de tiempo en que está accesible toda la información de la red.

  Se puede llegar a esta conclusión debido a que nuestro sistema es un sistema hergódico (o sea, es una cadena de Markov con espacio de muestras irreductible), lo cual no está dado por ninguna propiedad particular del sistema sino por el modelo que nos definimos al encarar el problema.

- Por otro lado, una confiabilidad $R_{s,V}$ a su vez indica:
    - la probabilidad de que un nodo cualquiera tenga acceso a toda la información, o análogamente
    - la probabilidad de que un nodo no va a recibir un falso rechazo (se le informa que la información no está disponible cuando en realidad sí lo está)

  Esta medida es de carácter más local a un nodo, mientras que la anterior es de carácter global a la red como un todo.

  Por último, se encontró una relación entre la confiabilidad $R_{s,t}$ y el valor calculado por la Fi de Porcentaje. Esta última define la probabilidad de que un nodo cualquiera encuentre cualquier información (es una idea de carácter aun más local que la confiabilidad $R_{s,V}$). Este valor coincide con el promedio de las confiabilidades $R_{s,t}$ variando s y t en el conjunto de todos los nodos del problema en cuestión.

Aplicaciones a entregar

Se definió entregar, dos aplicaciones, una con el visualizador, el evaluador y los traductores de GXL a Heidi y otra con el optimizador.

El evaluador y el traductor, serán accesibles desde el menú “Archivo” del actual visualizador. El evaluador tendrá dos casos de uso, sobre el grafo que se esta visualizando (se resuelve el problema general) y aplicado al problema particular de redes P2P (grafo completo con equiprobabilidad en las aristas, dado por las cantidad de conexiones promedio permitidas a cada nodo).

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Comenzar el diseño de la aplicación de optimización.
- Avanzar con la prueba de validación de la aplicación de evaluación.
- Avanzar con el informe.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	90	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	10	Se esta evaluando el algoritmo a implementar
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Visualizador de Grafos	90	Restan detalles y mejoras menores, y terminar los diagramas de clases y de colaboración.
Traductores Heidi a GXL y GXL a Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Manuel Rodríguez	21 de Octubre de 2004	Autor de la versión inicial.
Diego Costanzo	23 de Octubre de 2004	Modificaciones y aprobación
Jorge Coll	23 de Octubre de 2004	Modificaciones y aprobación

Minuta de Reunión #14 - 16/11/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Martes 16 de Noviembre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Optimización: Analizar los resultados obtenidos hasta el momento y evaluar líneas de trabajo futuro.
- Evaluación: Pruebas de comparación del tiempo en C++ contra Java.
- Definición de tareas para la próxima reunión de avance.

Desarrollo de Temas:

Optimización: Analizar los resultados obtenidos hasta el momento y evaluar líneas de trabajo futuro.

Realizando pruebas con la herramienta de Optimización, se detectó que bajo ciertas condiciones existe un óptimo trivial del problema que, considerando el espacio de soluciones como una matriz en la que las filas varían los distintos valores de C, y en las columnas los valores de D, que se encuentra en la posición (N-1, 1) siempre que el límite de carga (LC) sea mayor que N*(N-1).

Esta solución, así como todas las que tienen un C cercano a N, en la práctica son inviables ya que implicarían permitir conexiones desde un nodo a prácticamente el resto de los nodos. Es por esto que se decidió excluir la fila C = N-1 del espacio de soluciones y además pedir como parámetro de entrada un máximo C, para limitar hasta donde se quiere buscar la solución.

En este sentido se barajaron varias posibilidades:

- Utilizar combinaciones de valores de N y C más acordes con la realidad, por ej., N = 10000 y C = 50. Esta variante es poco práctica para la aplicación desarrollada, ya que las pruebas muestran tiempos de ejecución inviables para parámetros de estos órdenes.
- Afinar la evaluación de la confiabilidad: todos los puntos del espacio de problemas (menos la fila C= N-1) tienen confiabilidad menor que 1. Muchos de ellos nos dan 1 por problemas de precisión tanto en el nro. de muestras que se toman para el Monte Carlo como por problemas de overflow en la representación interna de datos de la máquina. Para ver que tan bueno son los valores que estamos retornando se propuso calcular y desplegar el sigma de la confiabilidad que estamos calculando.

- El propio método está demostrando que el óptimo tiene una regularidad a estar en una zona determinada del espacio de problemas, lo cual indicaría la necesidad de realizar un estudio analítico para intentar buscar una explicación más formal a este comportamiento. Esto escapa al alcance de nuestro proyecto, por lo que quedaría marcado como trabajo futuro.

Continuando con el estudio de estas primeras informaciones que nos presentan los resultados obtenidos hasta el momento, se realizaron algunas sugerencias para continuar con el análisis de los resultados de nuestro problema de optimización, entre las que mencionamos las siguientes:

- En lugar de parar el procesamiento cuando se encuentra un óptimo, realizar una búsqueda completa y retornar el conjunto de puntos de nuestra matriz de problemas que forman los vértices de los bloques de la frontera de carga y las confiabilidades en ellos.
- Tratar de ver la influencia de la carga en los resultados, por ejemplo, ver como cambia la frontera al cambiar el valor del parámetro de carga manteniendo fijo el parámetro de cantidad de nodos.
- Se podría llegar a hacer un estudio del espacio de soluciones al ir variando N y el límite de carga en la misma proporción (se notó en la reunión que en estas condiciones la frontera de cargas no variaría demasiado, mientras que si se podría sacar algún resultado interesante de la variación de la frontera de confiabilidad) aunque dado el tiempo que insumirían estas pruebas, las mismas quedan en stand-by.
- Ver que pasaría al definir un punto de corte arbitrario en C y reducir el espacio de búsqueda a las filas comprendidas entre 1 y el punto de corte. Intuitivamente, si definimos el valor de corte de C como C_c, un óptimo del problema estará en el punto (C_c, 2) o en el “escalón” más cercano a la línea de corte de la frontera de carga, cosa que sería interesante de comprobar por lo menos de manera empírica.
- No aportaría mucho al análisis que estamos haciendo (que es básicamente un análisis del algoritmo) usar valores de N muy grandes, ya que la forma de la frontera y el comportamiento de la aplicación seguramente no cambien demasiado.

En general, sería bueno comenzar por definir exactamente que es lo que pretendemos de las pruebas que realicemos, siendo en este sentido una opción más que interesante, el hacer un análisis del comportamiento de nuestro algoritmo.
También podría ser de utilidad para las pruebas, definir que significa un valor determinado para la carga (por ej. en una conexión de ancho de banda B se podrían enviar x paquetes) y en este marco analizar que valores serían razonables para nuestros parámetros.

En definitiva, se definió realizar una serie de pruebas para un N = 50, variando el límite de carga en el siguiente conjunto de valores {20.000, 200.000, 2.000.000, 20.000.000} y una corrida para un N = 1000 y C = 400.000 donde esperamos que se comporte en forma parecida al caso N = 50 C = 20.000

Evaluación: Pruebas de comparación del tiempo en C++ contra Java.

Primero que nada definimos el objetivo de las pruebas, el cual sería simplemente comparar el rendimiento de los diferentes lenguajes para nuestro tipo de aplicación.
Esto implicaría inicialmente hacer el “mejor” código para la aplicación teniendo en mente las propiedades de cada uno de los lenguajes, lo cual ya se realizó.

Algunas consideraciones para la realización de las pruebas para la comparación de la performance en ambos lenguajes:

- La diferencia de performance entre los lenguajes bajo las condiciones planteadas debería ser lineal respecto al nro. de casos que se consideren.
- Una idea para hacer las pruebas es tomar el conjunto de parámetros variables N, C, D y Fi, e ir fijando de a tres de ellos y variando el restante.

- En principio hacer las pruebas usando el algoritmo de Monte Carlo (justificando que es el que vamos a seguir usando para la parte de optimización y que la Generación Completa de Estados es viable solo para N<7).

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:

- Realizar las pruebas del Optimizador, para el juego de parámetros mencionado arriba: N = 50 y límite de carga = {20.000, 200.000, 2.000.000, 20.000.000}
- Cambiar el Optimizador para retornar el conjunto de valores óptimos.
- Avanzar con las pruebas de comparación de la eficiencia del Evaluador en C++ y Java.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	90	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	70	Está implementada la funcionalidad básica, faltando realizar pruebas de la misma, completar detalles y hacer la documentación.
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Visualizador de Grafos	90	Restan detalles y mejoras menores, y terminar los diagramas de clases y de colaboración.
Traductores Heidi a GXL y GXL a Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Manuel Rodríguez	17 de Noviembre de 2004	Autor de la versión inicial.
Manuel Rodríguez	18 de Noviembre de 2004	Correcciones en la redacción.
Jorge Coll	20 de Noviembre de 2004	Lectura, modificaciones y aprobación
Diego Costanzo	21 de Noviembre de 2004	Lectura, modificaciones y aprobación

Minuta de Reunión #15 - 22/11/2004
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Martes 22 de Noviembre de 2004
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Evaluación: Comentarios sobre las pruebas de validación.
- Optimización: Comentarios sobre las conclusiones preliminares obtenidas.
- Definición de tareas para la próxima reunión de avance.

Desarrollo de Temas:

Evaluación: Comentarios sobre las pruebas de validación.

Se estuvieron comentando los resultados parciales obtenidos para las pruebas de validación de la aplicación de Evaluación. Algunos puntos interesantes que surgieron de estos comentarios son los siguientes:

- Para el caso particular N = 3 y C = 2, la fórmula de carga que estamos usando como estimativo de la misma no está definida. Habría que evaluar más en detalle el por qué de este comportamiento en este punto particular, aunque seguramente sea un problema en el desarrollo de la serie a partir de la que se generó la fórmula.

- Se notó que la evaluación de la carga utilizando el método de Mote Carlo Antitético con D = 1 la cantidad de aristas sorteadas en cada caso coincide siempre con la cantidad media de aristas del grafo (por la construcción del método) y ésta última coincide siempre con la cantidad de paquetes (ya que D = 1).
  Esto explica el comportamiento observado en algunas pruebas realizadas.

- La fórmula para la carga tiende a funcionar mejor cuando C aumenta, ya que la dispersión de las aristas tiende a disminuir.

- Algunas medidas interesantes que se podrían llegar a realizar, como forma de evaluar la "calidad" de los mismos, son el cociente de los sigmas de los métodos aproximados, así también como el cociente de los tiempos de respuesta de cada uno.

Una medida que englobaría todas estas mediciones de la calidad sería la Eficiencia Relativa, definida como el producto del cociente de los sigmas y el cociente de los tiempos.

- En la comparación de la eficiencia de los métodos implementados en C++ y en Java, aparentemente el único parámetro que afectaría los resultados sería la Carga. En caso de confirmarse esto, se podrían hacer algunas pruebas más para confirmar esta tendencia inicial.

Comentando sobre los cálculos que ya se habían hecho para la comparación de tiempos entre los métodos, los mismos no pierden validez a pesar de haberse mejorado la fórmula de carga desde el momento de su realización.
Como una cuestión a nivel general, se formatearán los resultados presentados en las planillas y en las gráficas de forma de adaptarlos mejor a lo que será el formato de informe final.

Optimización: Comentarios sobre las conclusiones preliminares obtenidas.

Las conclusiones planteadas, salvo la corrección de que el valor obtenido para la carga es exponencial en D y polinominal en C, son correctas. Habría que hacer un desarrollo más formal y detallado de las mismas de forma de poder incluirlas en el informe final.

Sobre algunas líneas de trabajo futuro que surgen de nuestro proyecto, algunas de ellas serían:
- Encontrar una fórmula general para el cálculo de la confiabilidad en cada punto.
- Buscar alguna justificación para el buen comportamiento de la fórmula de carga utilizada.

También se planteó la posibilidad de realizar algún otro análisis en busca de más conclusiones del trabajo, entre las que mencionamos:
- Buscar y analizar un caso concreto real (una red de N nodos con ancho de banda B, etc.)
- Hacer un análisis similar al que ya se hizo para otra de las funciones objetivo, por ej. el porcentaje de nodos conectados por caminos largo menor o igual a D.

El primero de estos puntos se descartó debido al tiempo que insumiría, siendo inviable a esta altura del proyecto; mientras que el segundo será considerado en función de los avances que se vallan produciendo de aquí en más.

Definición de tareas para la próxima reunión de avance

Se definió para la próxima reunión, avanzar en las siguientes:

Tareas:
- Terminar con las pruebas de validación del Evaluador y el Optimizador.
- Poner a punto las diferentes aplicaciones que se han venido desarrollando a lo largo del proyecto.
- Avanzar en el desarrollo del informe final.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	90	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	75	Está implementada la funcionalidad básica, faltando realizar pruebas de la misma, completar detalles y hacer la documentación.
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Visualizador de Grafos	90	Restan detalles y mejoras menores, y terminar los diagramas de clases y de colaboración.
Traductores Heidi a GXL y GXL a Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Manuel Rodríguez	23 de Noviembre de 2004	Autor de la versión inicial.
Jorge Coll	20 de Noviembre de 2004	Lectura, modificaciones y aprobación
Diego Costanzo	21 de Noviembre de 2004	Lectura, modificaciones y aprobación

Minuta de Reunión #16 - 2/02/2005
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Miércoles 2 de Febrero de 2005
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Establecer cronograma para el resto del proyecto
- Definir formato del informe
- Priorizar las tareas pendientes

Desarrollo de Temas:

Establecer cronograma para el resto del proyecto

Se definió como meta, realizar la defensa hacia fines de abril. En febrero, mes en que se le dedicadaza una carga horaria menor a la establecida, se decidió trabajar sobre los anexos (ver sección siguiente). En marzo elaborar el informe ejecutivo y cerrar todos los temas “chicos” pendientes. A principios de Abril tener muy cerca de la versión final el informe, para entregarlo a mediados de mes y una versión primaría de la presentación, para trabajar sobre ella en la segunda quincena.

Definir formato del informe

Se optó por hacer un informe con un resumen ejecutivo, que son los puntos 1 al 7 de la siguiente propuesta de índice y luego ampliar cada aplicación en un anexo para cada una, describiendo el diseño, la implementación y las conclusiones. Como anexo también estará el manual del usuario, las actas y los cronogramas y la biblografía.

1. Introducción
2. Objetivos
3. Visualizador
4. Traductor
5. Evaluador
6. Optimizador
7. Conclusiones y trabajos futuros

Anexos

- Estudio de Grafos

- Visualizador
  - Motivación, objetivos, alcance
  - Modelo conceptual
  - Diagramas de colaboración
  - Diagramas de clases
  - Arquitectura
  - Pruebas de validación
  - Conclusiones
- Traductor
- Evaluador
  - Motivación, objetivos, alcance
  - Algoritmos
  - Modelo conceptual
  - Diagramas de colaboración, pseudocodigo
  - Diagramas de clases
  - Pruebas de validación
  - Conclusiones
- Optimizador
  - Motivación, objetivos, alcance
  - Algoritmos
  - Modelo conceptual
  - Diagramas de colaboración, pseudocodigo
  - Diagramas de clases
  - Pruebas de validación
  - Conclusiones
- Manual de Usuario
- Cronograma, Actas
- Bibliografía

Asignar prioridades a tareas pendientes

Se reviso la lista de tareas pendientes, asignándole mayor prioridad a las relacionadas con el informe, las que ya tenían un porcentaje elevado de realización y la de hacer las aplicaciones accesibles desde un servidor web. Se descartaron algunas tareas menores que no aportaban demasiado, así como la realización de un testing formal.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	90	Básicamente resta, terminar informe de validación y terminar documentación.
Aplicación de Optimización	75	Está implementada la funcionalidad básica, faltando realizar pruebas de la misma, completar detalles y hacer la documentación.
Aplicación para leer y visualizar grafos en formato Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7
Visualizador de Grafos	90	Restan detalles y mejoras menores, y terminar los diagramas de clases y de colaboración.
Traductores Heidi a GXL y GXL a Heidi	95	Restan solo detalles y mejoras menores. Ver planilla de tareas v 1.7

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	7 de marzo de 2005	Autor de la versión inicial.
Manuel Rodriguez	10 de marzo de 2005	Lectura y Aprobación
Jorge Coll	10 de marzo de 2005	Lectura y Aprobación

Minuta de Reunión #17 - 29/03/2005
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Martes 29 de Marzo de 2005
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Comentar los avances y formular algunas consultas
- Comentar la documentación generada hasta el momento.
- Definir Cronograma de Reuniones y Entrega

Desarrollo de Temas:

Comentar los avances y formular alguna consulta

Se comento en cuanto a la documentación se tienen terminada la primera versión de los Anexos y una parte de la primera versión del informe ejecutivo. En cuanto a la evaluación se redacto el análisis de los resultados y se agregaron algunas validaciones. También se arreglo la función de cifras significativas y se moró la performance la la Fi de CDA y Fi Porcentaje
En cuanto al optimizador se lo dividió en capas, se mejoro la salida (según acta anterior) y se hicieron los javadoc.
Asimismo se hicieron los instaladores de los las tres aplicaciones y el manual de usuario del evaluador.

Se hicieron consultas, como por ejemplo ver por qué en el caso 6 el MCA da sigma 0, se confirmo que los NAN son por un problema de represtanción de punto flotante, se evaucaron dudas acerca de los nombres de la graficas y sobre el valor de la confiabilidad de UDL.

Comentar la documentación generada hasta el momento.

Se hizo una lectura de la primera parte del menú ejecutivo y se comento que faltaría una sección de Resumen y palabras claves, nombrar los apéndices en la introducción y quitar el desarrollo de las 3 formulas en esta sección dejando solo el de la formula utilizada finalmente.

Definir Cronograma de Reuniones y Entrega

Se fijaron reunión para los artes 5 de Abril, 12 y 19 de Abril. Para la primera la idea es llevar la primera versión del ejecutivo, para la segunda el ejecutivo terminado y la primera versión de la presentación y para la versión final de paliaciones e informe final para su entrega.

La entrega se definió que fueran 5 copias, una para cada miembro del tribunal, otra para la biblioteca y otra para el tutor y 2 CD. Se manejo como fecha tentativa para la defensa el jueves 29 de abril.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	98	Poner nombre de la aplicación e iconos, crear instalador.
Aplicación de Optimización	98	Arreglar caso c=1, la optimización para porcentajes, poner nombre a la aplicación e iconos y crear instalador
Visualizador de Grafos	95	Restan incorporarle última versión del evaluador, poner nombre a la aplicación e iconos, crear instalador y hacerlo ejecutable desde la web.
Traductor GXL-Heidi	95	Solucionar algunos bugs, terminar de documentar pruebas de validación, poner nombre a la aplicación e iconos

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	2 de Abril de 2005	Autor de la versión inicial.
Manuel Rodriguez	5 de Abril de 2005	Lectura y aprobación
Jorge Coll	5 de Abril de 2005	Lectura y aprobación

Minuta de Reunión #18 - 7/04/2005
Proyecto: Evaluación y Optimización de Confiabilidad Diámetro Acotada.

Datos de la Reunión:

Fecha: Jueves 7 de Marzo de 2005
Lugar: Facultad de Ingeniería – INCO
Asistentes:

Nombre	Rol
Héctor Cancela	Tutor
Jorge Coll	Miembro del Grupo
Diego Costanzo	Miembro del Grupo
Manuel Rodríguez	Miembro del Grupo

Agenda:

- Comentar los avances y formular algunas consultas
- Leer las correcciones y sugerencias de la documentación
- Definir tareas para la próxima reunión de avance.

Desarrollo de Temas:

Comentar los avances y formular algunas consultas

Se le comunico al tutor que nos pareció interesante su propuesta de presentar los resultados del proyecto en alguna conferencia y se mostró un bosquejo rápido de lo que sería su contenido. Quedamos que luego de elegida la misma, la cual posiblemente sea la conferencia anual de informática que se desarrolla en la provincia de Entre Ríos (Argentina), nos pondríamos a trabajar en el tema.

Comentamos que se logró que las aplicaciones sean ejecutables a través de web usando la tecnología Java Web Start que es la implementación del JNLP (Java Network Launching Protocol) de Sun. En este sentido quedamos de enviarle una primer versión del la pagina del proyecto para más que nada asegurarnos de que funciona la inicialización de las aplicaciones en el servidor de facultad. Se definió que la página del curso sea la siguiente:
www.fing.edu.uy/inco/grupos/invop/cda2005

Se le entrego una nueva versión del Informe Ejecutivo (vía mail el martes 5), todos los manuales de usuario excepto el del optimizador y se comento se avanzo en las aplicaciones mejorando la interfaz grafica, solucionando bugs en el traductor y el Optimizador, quedando todas las aplicaciones prontas a excepción de calculo de la carga para C=1 en el optimizador, enganchar el evaluador al visualizador, asignarle un nombre a las aplicaciones, poner iconos en los botones y diseñar un logo para cada aplicación.

Se realizaron varias consultas como ser:

Como manejar el caso C=1 en la función que calcula el D, para determinar la frontera, lo cual quedo aclarado.
Se confirmó que el resultado de confiabilidad para la red UDL es correcto
Aclaramos una duda acerca de cómo variaba la cantidad de estados respecto al la cantidad de nodos y la carga respecto a CCP y D.
Hablamos de la aplicación de nuestro modelo a las redes P2P, en este sentido definimos que los grafos de nuestro modelo son arbitrarios y las confiabilidades de las aristas pueden ser distintas entre sí. De hecho los algoritmos de evaluación implementados, evalúan grafos así. Sin embargo inspirados por las redes P2P trabajaremos con un caso particular de ese modelo donde la cantidad de nodos representa la cantidad de nodos activos de la red P2P y la confiabilidad de las aristas esta dada por la cantidad de conexiones promedio que un nodo conoce CCP sobre la cantidad de nodos activos menos uno.
También volvimos a ver el razonamiento para la formula de carga 3. La idea central fue ver que un nodo j cualquiera va a estar conectado N-2 vecinos distintos del nodo i desde donde le viene el paquete y como cada arista tiene probabilidad C/(N-1) ese nodo j estará contado a N-2*C/(N-1) nodos y por tanto enviara esa cantidad de paquetes.
Se definió que las planillas con los resultados de la prueba de validación y análisis del evaluador y el optimizador estuvieran disponibles en la página del proyecto y en el CD, pero no incluirla como anexo en el informe.
Acerca de la impresión del proyecto se nos sugirió hacerla doble faz.
El hecho que MCA a veces diera mayor varianza que MCC nos explicó que es debido al hecho que MCC es optimista para confiabilidad cerca de 1.
También comentamos que un resultado conocido que en la literatura y que sucede también en nuestro trabajo es la rápida transición de fase de la confiabilidad al aumentar la CCP o el D.

Leer las correcciones y sugerencias de la documentación

Se leyeron todas las sugerencias y correcciones del anexo del evaluador, del optimizador del traductor y del visualizador.

Definir tareas para la próxima reunión de avance

Se definió para la próxima reunión hacer:

- Una primera versión de la presentación
- Enviar la primer versión de la pagina web del proyecto
- Plasmar las correcciones y arreglos en la documentación indicados hoy.

Status de Aplicaciones:

Tarea	Status (%)	Comentarios
Aplicación de Evaluación	99	Poner nombre de la aplicación e iconos
Aplicación de Optimización	98	Arreglar caso c=1, la optimización para porcentajes, poner nombre a la aplicación e iconos.
Visualizador de Grafos	98	Restan incorporarle última versión del evaluador y poner nombre a la aplicación e iconos.
Traductor GXL-Heidi	98	Terminar de documentar pruebas de validación, poner nombre a la aplicación e iconos

Historia de revisiones

Actuante	Fecha	Comentarios
Diego Costanzo	18 de Abril de 2005	Autor de la versión inicial.
Jorge Coll	19 de Abril de 2005	Lectura y Aprobación
Manuel Rodriguez	19 de Abril de 2005	Lectura y Aprobación