UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

ESTUDIO DE LA LOCALIZACIÓN ÓPTIMA DE REFLECTORES DE RUTAS BGP

CARLOS ALBERTO PARÍS MURILLO

Julio de 2011

ESTUDIO DE LA LOCALIZACIÓN ÓPTIMA DE REFLECTORES DE RUTAS BGP

AUTOR: Carlos Alberto París Murillo

TUTOR: Luis F. Lago Fernández

Grupo de Neurocomputación Biológica (GNB)

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Julio de 2011

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi tutor, Luis, su esfuerzo, dedicación y paciencia en la realización de este Proyecto Fin de Carrera, gracias por amoldarte a mis cambios de ritmo en el trabajo, por estar siempre disponible…gracias.

En segundo lugar, debo agradecer el apoyo de la empresa Juniper Networks, que nos ha aportado los datos necesarios para la realización de este proyecto.

Gracias a mis amigos de siempre, tres de los cuales me acompañaron en esta andadura (Sergio, Tomás y Vicente), y a mis amigos de la universidad, Pablo, Low, Antonio, Los Katas (sé que me dejo gente pero el espacio es limitado), porque cuando se pasa tanto tiempo juntos es inevitable que se les considere ya amigos y no compañeros.

Y por supuesto gracias a toda mi familia, a mi padre por su infinita paciencia, saber que siempre estaba ahí para lo que quisiera tranquiliza mucho, a mi madre por su infinita impaciencia, sólo ella sabe cómo ponerme las pilas, a Raúl por la cantidad de veces que me ha cerrado la puerta de mi habitación para no molestarme mientras estudio, soy lo que soy gracias a vosotros.

Gracias a Violeta que me ha acompañado durante todos estos años, aguantando mis tiempos de estudio y ocio, gracias por apoyarme en todo momento y por todo el cariño que me has dado. A mi abuela por todas las novenas rezadas para que yo aprobase, a mi tío por confiar siempre en mí aun cuando le decía que no habían salido bien las cosas. A mi abuelo.

Gracias.

RESUMEN

Este proyecto de fin de carrera estudia el posicionamiento óptimo de los reflectores de ruta BGP en redes de ordenadores con gran número de nodos, concretamente en redes de tipo Tier-1. En este tipo de redes las soluciones de tipo mallado completo (full-mesh) no escalan bien.

La primera parte del proyecto se ha dedicado a revisar la literatura para conocer las distintas aproximaciones que se han utilizado para resolver los problemas de escalabilidad en las redes actuales. Son dos las principales soluciones que se han propuesto en este sentido: las confederaciones y la reflexión de rutas. Esta última es el objetivo del presente proyecto.

Una vez revisado el estado del arte, se han implementado en Matlab algunos de los algoritmos que aparecen en la literatura para la configuración de redes BGP con reflexión de rutas. El primer hito alcanzado es un programa capaz de comprobar si una topología BGP dada opera de forma óptima, considerando "operar de forma óptima" como funcionar de la misma manera que lo haría una topología de tipo full-mesh. Este concepto es denominado fm-optimalidad.

El núcleo del proyecto consiste en el desarrollo de algoritmos para el diseño directo topologías BGP que sean fm-óptimas al tiempo que minimicen el coste IGP asociado a las sesiones iBGP entre los nodos de la red. Tomando como punto de partida el algoritmo de Buob et al. 2008 [1], se ha desarrollado un nuevo algoritmo que diseña la topología de manera incremental.

La evaluación de los dos algoritmos sobre un conjunto diverso de redes (incluyendo una de tipo tier-1) muestra que el nuevo algoritmo es más rápido que el original y que es capaz de encontrar soluciones comparables. Además, si se elige de manera apropiada el orden en el que se van incorporando los nodos, el nuevo algoritmo es capaz de proporcionar soluciones claramente mejores.

Finalmente se presenta una modificación del algoritmo que permite crear topologías fm-óptimas robustas frente al fallo de alguno de los nodos.

PALABRAS CLAVE

Reflector de ruta, fm-optimalidad, full-mesh, topología BGP, Tier-1

ABSTRACT

This M.Sc.Thesis studies the optimal BGP route reflectors' location in computer networks with a large number of nodes, specifically in Tier-1 networks. For these networks, the full mesh solution does not scale properly.

The first part of the project is dedicated to literature review in order to study the different approaches that have been proposed to solve the scalability problems in large networks. There exist two main solutions for this aim: confederations and route reflection. This project is focused on the last one.

After the state of the art review, we have implemented in Matlab some of the algorithms found in the literature designed to configure BGP route reflection topologies. The first achieved goal is a program that is able to check if a given BGP topology operates correctly, defining "correctly" as working in the same way that a full-mesh topology would do. This concept is known as fm-optimality.

Project`s core is the development of algorithms that are able to design fm-optimal BGP topologies while minimizing the IGP weight associated to the iBGP sessions established between the nodes. Departing from the algorithm in Buob et al. 2008 [1], we have developed a new algorithm that incrementally constructs the network topology.

The algorithms evaluation using several different networks (including a tier-1 network), shows that our new algorithm is faster than the one found in Buob et al 2008 [1], and that it is able to reach similar solutions. Furthermore, if the nodes are added in the right order, the new algorithm provides much better solutions.

Finally, we present a modification to the algorithm that is able to add redundancy to our fm-optimal BGP topologies in order to make them robust against node failures.

KEY WORDS

Route reflector, fm-optimality, full-mesh, BGP topology, Tier-1

INDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

INDICE DE TABLAS

1

Introducción

Una red es un conjunto de nodos conectados por enlaces. Un ejemplo de red que todos conocemos es la red de carreteras, si la pudiéramos ver de forma aislada y a vista de pájaro, observaríamos que básicamente consiste en interconectar cruces a través de carreteras. Por ejemplo en la figura 1 podemos ver dos cruces conectados entre sí a través de una carretera (en realidad dos, una de ida y otra de vuelta)

Figura 1: Red de carreteras

En este proyecto fin de carrera, vamos a trabajar con redes de ordenadores. En este tipo de redes, los nodos son los ordenadores. Éstos se intercambian datos (en forma de paquetes IP) a través de los enlaces. Estas redes no son totalmente malladas, con lo que es posible que el origen de los datos no esté directamente conectado con su destino, por esta razón necesitamos algún método para orientar los datos en la red consiguiendo así que lleguen al destino deseado. Este método es el enrutamiento.

Llamamos enrutamiento al mecanismo que permite transportar el tráfico hacia cualquier punto de la red. En el caso de la red de carreteras, el transporte o enrutamiento, está controlado por las señales, semáforos o guardias de tráfico. Esto permite a los conductores saber qué camino seguir en la red de carreteras en función de su posición actual y en función de su destino. En las redes de ordenadores los protocolos de enrutamiento cumplen esta misión (decidir la ruta óptima entre 2 puntos, conociendo origen y destino). Las decisiones de enrutamiento se toman en los *routers*.

Un router es un dispositivo que distribuye paquetes de datos entre redes de ordenadores, cuando a un router le llegan paquetes que redirigir, consulta la información del paquete para determinar su destino final. Luego, utilizando esta información consulta la tabla de rutas para saber hacia dónde enviar el paquete. Este proceso se repite hasta llegar al destino del paquete.

Internet puede verse como un conjunto de redes de ordenadores interconectadas unas con otras. Algunas redes, llamadas redes backbone (o redes de núcleo), constituyen el esqueleto de Internet, es decir sus conexiones troncales. Estas redes se encargan de permitir el transporte de la información entre el conjunto de las redes de Internet. Este tipo de redes se componen de un gran número de routers comerciales, gubernamentales, universitarios y otros de gran capacidad interconectados que llevan los datos entre países, continentes y océanos del mundo.

En sus inicios, internet era una red unificada. Los routers intercambiaban su información de transporte a través de un único protocolo de transporte, el GGP (Gateway to Gateway Protocol). Cada router almacenaba en su tabla de rutas las distancias hacia todas las redes IP de Internet y la forma óptima de acceder a cada una de ellas. Esta configuración era idónea para la red de la época, pero conforme el tiempo iba pasando empezaban a surgir numerosos problemas. Uno de los más evidentes era que, debido al continuo crecimiento de la red, el tamaño de las tablas de rutas se estaba haciendo insostenible ya que crecía linealmente con la red. Además de esto, el número de mensajes intercambiados por los routers también crecía enormemente.

Por esto mantener a internet como una red única se hizo inviable, con lo que se decidió subdividir esta única red en varios sistemas autónomos, o AS (Autonomous System) (véase figura 2). Debido a esta subdivisión, Internet hoy en día es un conjunto descentralizado de redes de comunicación interconectadas. Un AS es una porción de red administrada por un mismo organismo, como por ejemplo, un operador de Internet o una empresa. Internet hoy está constituida por más de 28.000 AS. Cada AS se identifica por un número llamado ASN (AS number) el cual podemos ver representado en la figura 2.

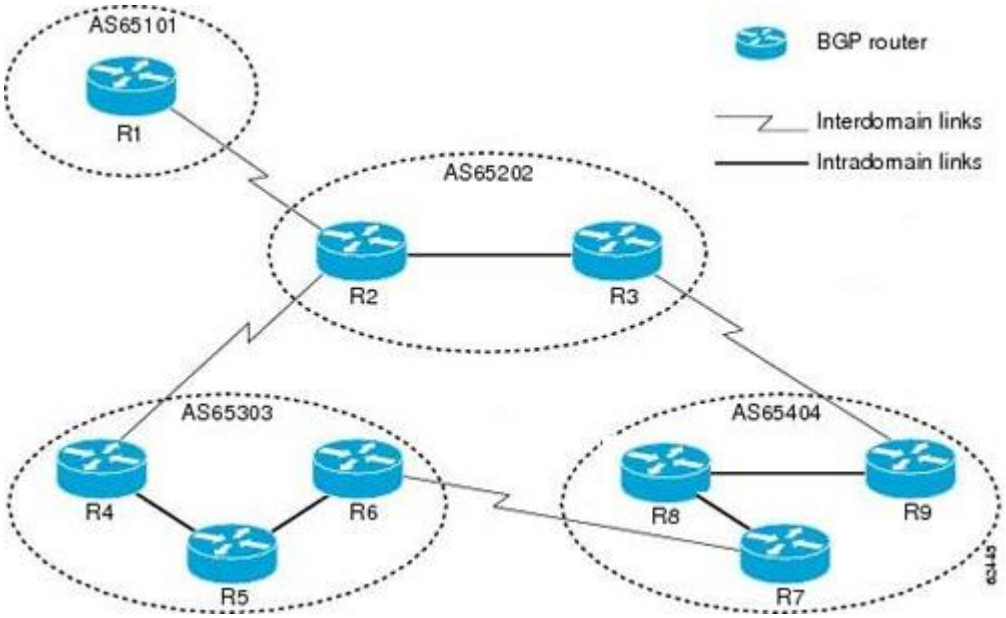


Figura 2: Sistemas Autónomos (AS)

De subdividir Internet en sistemas autónomos se derivan dos tipos de protocolos de transporte:

- Los protocolos de tipo IGP (Interior Gateway Protocol), que se encargan esencialmente del transporte hacia los destinos internos al AS.

- Los protocolos de tipo EGP (Exterior Gateway Protocol), que garantizan el traspaso de la información de enrutado entre los distintos AS, y que lo difunden en cada AS. Estos protocolos permiten a los routers saber cómo enrutar los destinos exteriores a su sistema autónomo.

En la figura 3 podemos ver el marco de acción de los protocolos IGP y EGP. Como hemos mencionado IGP actúa en las comunicaciones internas al AS y EGP en las externas a los sistemas autónomos. En la imagen podemos observar como conectamos dos AS a través de una conexión EGP.

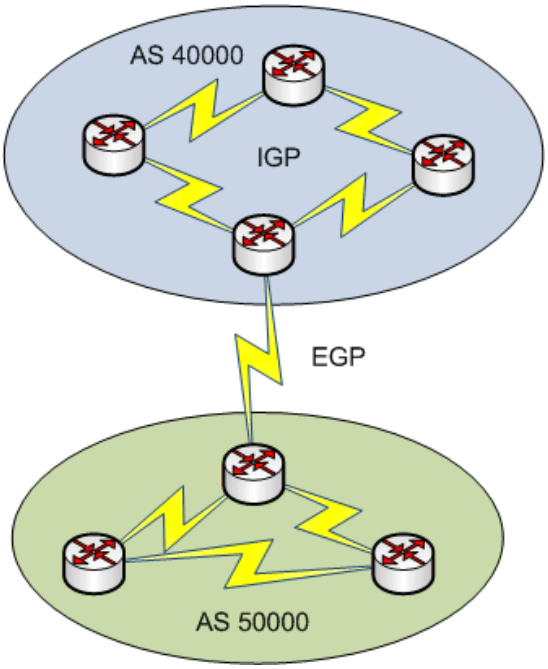


Figura 3: IGP vs BGP

La combinación de estos dos tipos de protocolos permite enrutar tanto destinos internos al AS como destinos externos. En la actualidad el protocolo EGP utilizado en todo internet es el BGP-4. En cuanto al IGP, no podemos concluir que haya un único protocolo, puesto que debido a la autonomía de los AS, cada operador es libre de escoger el protocolo IGP que va a utilizar. El protocolo IGP que escoja será invisible fuera del AS.

Este proyecto fin de carrera se focaliza en el protocolo BGP, en particular, nos centramos en la optimización de las topologías iBGP que permiten a un AS enrutar correctamente direcciones obtenidas por eBGP.

1.1 Motivación

Como hemos comentado en la introducción. Internet consiste en un conjunto de dominios interconectados que se denominan Sistemas Autónomos (AS). Dentro de cada AS se utilizan protocolos IGP (Interior Gateway Protocol) para determinar las rutas internas óptimas. Entre dominios vecinos, la información sobre las rutas se intercambia utilizando el protocolo Border Gateway Protocol (BGP). Se establecen sesiones BGP externas (eBGP) entre routers pertenecientes a AS distintos, y sesiones BGP internas (iBGP) entre routers pertenecientes a un mismo AS. Cuando un router aprende, mediante eBGP, una ruta para alcanzar un determinado prefijo externo, debe comunicar esta información al resto de routers de su AS utilizando iBGP. En la configuración más tradicional, todos los routers del AS mantienen sesiones iBGP con todos (configuración *full-mesh*). De este modo un router conoce todas las posibles rutas para alcanzar un determinado prefijo, y puede elegir la mejor de todas ellas. Este tipo de configuración es válida para AS de poco tamaño, pero no escala bien con el número *n* de routers (el número de sesiones iBGP crece de manera cuadrática con *n*), y por tanto no es válida para AS muy grandes. Para conseguir una buena escalabilidad en estos casos se utilizan alternativas como las *confederaciones* o la *reflexión de rutas*, siendo esta última la solución más frecuente.

La reflexión de rutas añade una capa de jerarquía a la red en la que cada reflector sirve a un subconjunto de routers (clientes). La reflexión de rutas reduce significativamente el número total de sesiones iBGP en la red y mejora la escalabilidad. Sin embargo, como el protocolo BGP establece que un router sólo envía la información de su mejor ruta, la presencia de reflectores hace que determinadas rutas puedan quedar ocultas para determinados routers. Esto puede llevar a que algunos routers utilicen rutas sub-óptimas para alcanzar determinados prefijos, pues la mejor ruta para el reflector puede no coincidir con la mejor ruta para alguno de sus clientes.

Como vemos, las topologías *full-mesh* permiten que los routers seleccionen siempre la mejor ruta, pero escalan mal con el número de routers en el AS. Por otra parte, el uso de reflectores de rutas mejora la escalabilidad de la red, pero puede llevar a que algunos reflectores seleccionen rutas sub-óptimas. Recientemente en la literatura se ha introducido

el concepto de *fm*-optimalidad **[6]**. Se dice que una topología iBGP es *fm*-óptima si cada router utiliza las mismas rutas que utilizaría con una configuración de tipo *full-mesh*. Una topología iBGP que utilizase reflectores manteniendo la *fm*-optimalidad permitiría mejorar la escalabilidad garantizando la selección de las mejores rutas en todos los casos. En la literatura se han propuesto algoritmos tanto para la comprobación de la *fm*-optimalidad de una determinada topología iBGP (**[6]**), como para el diseño directo de topologías *fm*-óptimas que minimizan el coste IGP asociado a las sesiones BGP (**[1]**). Dichos algoritmos son el punto de partida para este proyecto.

1.2 Objetivos

En términos generales, el objetivo de este *Proyecto Fin de Carrera* es el estudio e implementación de algoritmos para el diseño de topologías iBGP *fm*-óptimas, la validación de dichos algoritmos con redes de tamaño pequeño o medio y la aplicación final de los mismos algoritmos a una red real de tipo *tier-1*. A continuación se detallan los pasos/sub-objetivos que abordamos:

1. Estudio de la literatura relacionada para analizar las soluciones que ya han sido consideradas.
2. Implementación en Matlab de algoritmos para la comprobación de la *fm*-optimalidad de una topología iBGP. Se partirá del algoritmo descrito en *Buob et al. 2007* **[6]**.
3. Análisis y validación de los algoritmos implementados utilizando redes simuladas de pequeño tamaño. El uso de redes pequeñas permitirá realizar un análisis "visual" de las redes para verificar el correcto funcionamiento estos algoritmos.
4. Implementación en Matlab de algoritmos para el diseño directo de topologías *fm*-óptimas. Como punto de partida se tomará el algoritmo descrito en *Buob et al. 2008* **[1]**.
5. Análisis y validación de los algoritmos implementados utilizando redes de tamaño pequeño y medio Dicha validación incluirá:
   - 5.1 Verificación de la fm-optimalidad de la topologías diseñadas.
   - 5.2 Comparación con el full mesh en base al número de sesiones iBGP instaladas y al coste IGP asociado a las mismas.

6. Aplicación de los algoritmos implementados a una red de grande de tipo tier-1 y análisis de resultados.

1.3 Organización de la memoria

La memoria está dividida en cinco capítulos:

El primero de estos capítulos sirve a modo de introducción general, se presentan los objetivos, las motivaciones, y por tanto se explica el porqué de la realización de este Proyecto Fin de Carrera, así como la organización del mismo.

En el segundo capítulo, llamado Estado del Arte, se establece el marco teórico en el que nos encontramos, explicando con algo más de detalle los protocolos utilizados actualmente en internet, además de esto en este capítulo exponemos de forma más precisa el problema que nos ocupa y las soluciones que podemos encontrar en la literatura

El tercer capítulo, titulado Diseño y Desarrollo, podemos considerarlo el núcleo del proyecto, en él mostraremos los programas que hemos desarrollado para alcanzar los objetivos y explicaremos aspectos esenciales para entender los resultados.

El cuarto capítulo es el más extenso del proyecto, en el se muestran los resultados obtenidos aplicando los algoritmos desarrollados a ejemplos concretos de redes.

En el quinto capítulo se exponen las conclusiones de este Proyecto Fin de Carrera, y posibles continuaciones del trabajo realizado.

Por último se presentan una serie de anexos en las que entre otras cosas incluiremos los programas implementados.

2

Estado del Arte

2.1 EL protocolo IP

Hoy en día Internet se basa en el protocolo IP (internet protocol). Gracias él, es posible identificar de manera universal un equipamiento informático accesible en la red Internet. En concreto, cada equipamiento (router, terminal…) conectado a Internet se define por una o más direcciones IP públicas. Los datos en una red basada en IP son enviados en bloques conocidos como paquetes. En estos paquetes, viene indicada la dirección IP del destinatario y la del remitente (ver figura 4).

La figura 4 muestra como está organizado un paquete IP, podemos observar que vienen indicadas las direcciones de origen y destino como habíamos mencionado

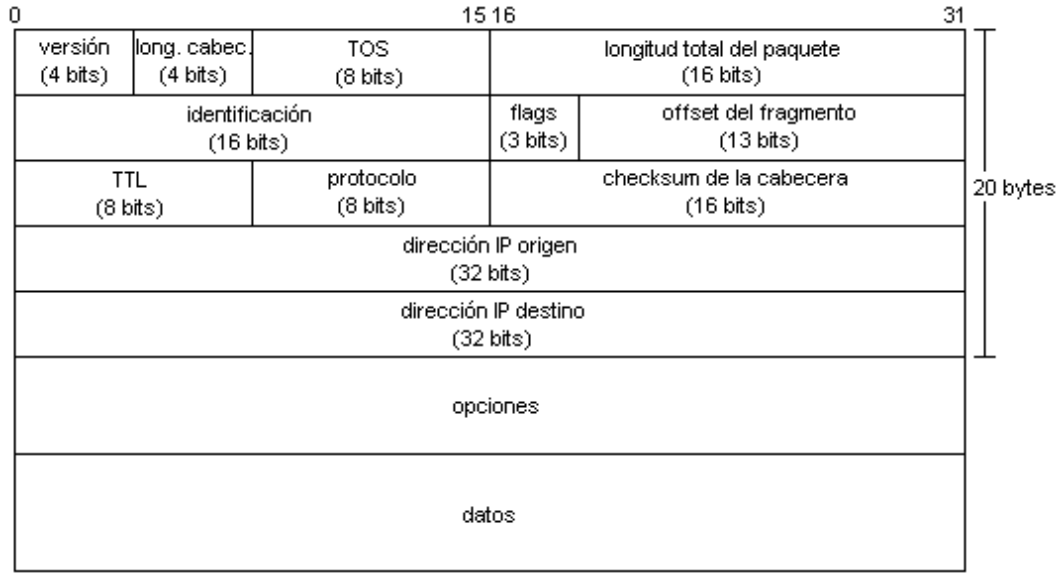


Figura 4: El paquete IP

Cuando un router recibe un paquete IP, examina su dirección de destino y lo redirige hacia el router vecino más adecuado en base a su tabla de rutas. El paquete repite este proceso sucesivamente hasta que llega a su destino.

Un router suele estar conectado a varios routers vecinos. Cuando recibe un paquete debe simplemente redirigirlo hacia uno de estos router vecinos, concretamente hacia el que considere más adecuado. Esta decisión se toma gracias a la tabla de rutas, en la que se asocian direcciones de destino con los router vecinos. Si volvemos a recurrir a la analogía entre red de ordenadores y red de carreteras, un router puede verse como un cruce de

carreteras, y una ruta (de la tabla de rutas) puede verse como una señal de tráfico (ver figura 5). Si un router recibe un paquete que no sabe dirigir (es decir, un paquete destinado a una dirección para la cual no tiene ruta) lo desecha.

Figura 5: Señales de tráfico, analogía con las rutas de la tabla de rutas

Con el fin de garantizar el transporte entre todo punto de la red, bastaría con que cada router tuviera en su tabla de rutas una ruta hacia cada destino de Internet. Sin embargo, el número de destinos en Internet es muy elevado, a pesar de que es posible agrupar bloques de direcciones IP contiguos con el fin de reducir el número de rutas. Estos bloques de direcciones son llamados bloques prefijos.

Si se incluye una dirección IP en varios prefijos, se utiliza la ruta de prefijo más restrictiva. Por ejemplo, si un router pretende alcanzar el destino *10.20.30.40* y dispone de una ruta para los prefijos *10.0.0.0 /8* (cualquier dirección IP que comienza por 10) y *10.20.30.0 /24* (cualquier IP que comienza por 10.20.30), elegirá la segunda. Hoy en Internet existen más de 250.000 prefijos.

2.2 Los protocolos de transporte

Los protocolo de transporte especifican cómo se escribe la tabla de rutas de cada router, para que cada uno ellos esté en condiciones de enrutar correctamente los paquetes que recibe. Un protocolo de transporte (enrutado o encaminamiento) se utiliza para que los routers intercambien información acerca de las rutas que conocen, de esta forma un router puede rellenar su tabla de rutas utilizando uno o varios protocolos de transporte.

```
C:\ >route PRINT
===========================================================================
Lista de interfaces
0x1 ........................... MS TCP Loopback interface
0x70003 ...00 16 76 ac a7 6a .Intel(R) 82562V 10/100 Network Connection
===========================================================================
===========================================================================
Rutas activas:
Destino de red     Máscara de red        Gateway     Interfaz  Métrica
        0.0.0.0          0.0.0.0  172.16.255.254   172.16.1.2     1
      127.0.0.0        255.0.0.0       127.0.0.1    127.0.0.1     1
     172.16.0.0      255.255.0.0      172.16.1.2   172.16.1.2    20
     172.16.1.2  255.255.255.255       127.0.0.1    127.0.0.1    20
 172.16.255.255  255.255.255.255      172.16.1.2   172.16.1.2    20
255.255.255.255  255.255.255.255      172.16.1.2   172.16.1.2     1
Gateway por defecto: 172.16.255.254
===========================================================================
Rutas persistentes:
  Ninguna
C:\>
```

Figura 6: Tabla de rutas

La información intercambiada entre dos routers utilizando un mismo protocolo de transporte se traspasa a través de una sesión específica de este protocolo. Una sesión puede ser vista como un canal de comunicación establecido entre dos routers. Podemos distinguir dos tipos de sesiones atendiendo a la proximidad entre los routers, las sesiones *monohop,* en las que ambos routers son vecinos y las sesiones *multihop*, en las que los routers no son adyacentes pero si son capaces de enrutarse.

Los protocolos de transporte que tendremos en cuenta durante este proyecto se basan en el intercambio de mensajes de enrutamiento. Estos mensajes permiten a cada router mantener una tabla de rutas actualizada en función de los acontecimientos que ocurren en la red, es lo que llamamos enrutamiento dinámico. De esta forma, los routers están en

condiciones de adaptar sus rutas en función de una avería, una parada por mantenimiento, o a la inclusión de una nueva ruta en la red.

Una propiedad de estos protocolos es que son protocolos distribuidos, en los que cada router calcula de manera autónoma sus rutas en función de la información de transporte transmitida por sus vecinos (en comparación con los protocolos centralizados, para los cuales una estación centralizada impone una decisión a cada equipamiento satélite). Por ejemplo se puede definir una ruta por defecto por la que saldrán los paquetes cuyo destino no esté indicado en la tabla de rutas. Esta ruta por defecto la podemos observar en la figura 6 y curiosamente también podemos ver una "ruta" de este tipo en la señal de tráfico que mostramos en la figura 5 cuando indica el camino de "todas direcciones".

Mediante los protocolos de transporte podemos asociar a cada enlace de la red una métrica o peso (ver figura 6 en la columna de métrica), que caracteriza la "distancia" que separa dos routers. Este concepto de distancia es muy general y no se traduce necesariamente en una realidad geográfica. Además no podemos considerarlo como una "distancia" estática, ya que esta puede variarse con mensajes específicos de los protocolos que sirven para anunciar cambios en los enlaces. Estos cambios pueden ser necesarios en muchas situaciones, por ejemplo puede incrementarse el peso de un enlace que está cercano a la saturación, primando de esta forma la utilización de otros caminos y reduciendo así la utilización del camino saturado.

En los siguientes apartados hablamos de forma más concreta de los dos protocolos de transporte más utilizados.

2.3 El protocolo IGP

El protocolo IGP se encarga del transporte interno al AS. Un operador de red pondera cada enlace IGP con un valor numérico estrictamente positivo, el peso IGP. Este peso caracteriza la "longitud" de un enlace entre dos routers. La longitud de un camino IGP es igual a la suma de los pesos de los enlaces a lo largo de este camino.

El protocolo IGP permite a cada router calcular los caminos más cortos atendiendo a esta métrica para alcanzar cualquier destino de su red IGP. Para efectuar este cálculo, cada router IGP necesita recoger y comunicar información sobre la topología de su red IGP. Concretamente, cada router IGP anuncia a los otros routers de la red IGP los destinos que puede alcanzar y los pesos de sus enlaces. A partir de esta información construye y adapta su tabla de la red IGP. La tabla de red puede ser vista como un grafo orientado ponderado, por lo tanto es posible calcular los caminos más cortos para alcanzar cada destino de la red con algoritmos de la teoría de grafos, como por ejemplo el algoritmo de Dijkstra.

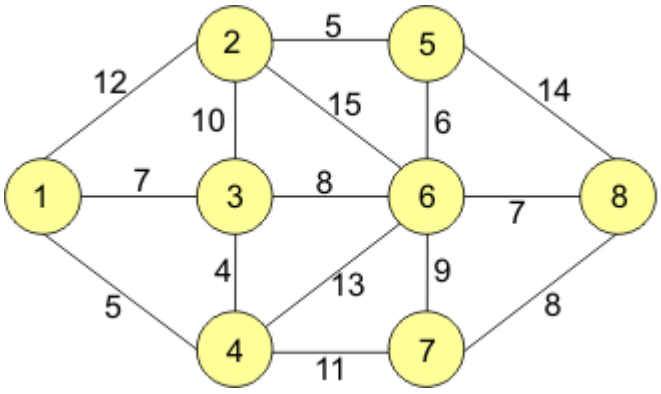


Figura 7: Grafo Ponderado

El protocolo IGP debe poder adaptarse a los cambios estructurales de la red IGP. Esto ocurre por ejemplo en una avería de un enlace o nodo, en la instalación de un nuevo equipamiento, o cuando el peso de un enlace es modificado. Estos cambios deben propagarse por toda la red para que cada router pueda actualizar su tabla de rutas. Cuando tiene lugar una actualización cada router debe recalcular sus rutas más cortas. Esta fase, durante la cual los routers IGP se adaptan a la estructura de la red y calculan sus

transportes, corresponde a la convergencia del protocolo IGP. Durante la convergencia IGP, pueden ocurrir bucles de transportes transitorios (llamadas micro cierres) lo cual podría implicar la pérdida de paquetes durante un corto plazo de tiempo (**[13]**).

Elección de los pesos IGP

El peso IGP de un enlace no caracteriza necesariamente la distancia geográfica que separa los dos routers conectados por dicho enlace. Además es típico que el enlace entre dos routers sea asimétrico, en otras palabras, el peso instalado entre un router *a* y un router *b* puede diferir del peso instalado entre *b* y *a*. Por ejemplo en la figura 8 se muestra que para ir de "uno" a "tres" la longitud es 3 y sin embargo para ir de "tres" a "uno" la longitud es 1. Ocurre una situación similar entre los routers "cinco" y "tres".

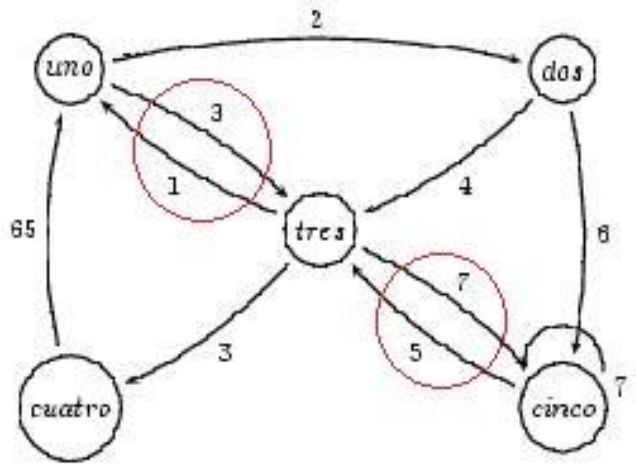


Figura 8: Interfaces Asimétricos

Cuando un router dispone de varios caminos de la misma métrica para alcanzar un destino, el router puede distribuir equitativamente su tráfico sobre estos caminos, haciendo de esta manera balanceo de carga (load balancing). Esta forma de actuar no es obligatoria, ya que el router puede configurarse para que elija un solo camino. Esto último puede hacerse de varias formas, por ejemplo de forma aleatoria o utilizando algún otro parámetro para decidir.

2.4 El protocolo BGP

El EGP consiste simplemente en asociar cada prefijo externo con la dirección IP de un router de borde del AS. Esta dirección debe ser por supuesto enrutable (por ejemplo por IGP) con el fin de ser utilizable. Esta dirección se llama gateway del AS o nexthop BGP (nomenclatura más común en la literatura). Una ruta BGP permite asociar un prefijo destino con un nexthop BGP. BGP-4 es el protocolo utilizado actualmente como EGP en todo internet.

Un router BGP ignora sistemáticamente las rutas relativas a un nexthop BGP que no sabe enrutar, en particular, si deja de saber cómo enrutar un nexthop borrará las rutas relativas a él. De esta forma las rutas BGP almacenadas siguen siendo coherentes con la estructura de la red. Una vez el nexthop BGP es elegido, el tráfico es conducido como si se destinara a este punto. Cada router repite esta operación sucesivamente para retransmitir el tráfico hasta hacerlo salir del AS. Si durante el camino, un paquete cruza routers que eligen otro nodo de salida, estaremos utilizando un camino erróneo.

Cada router puede corregir los atributos BGP que caracterizan a una ruta, por ejemplo puede dar un métrico al sentido BGP. Los atributos BGP permiten:

- Distinguir cuáles son las rutas más interesantes al sentido BGP
- Aplicar políticas de enrutado evolucionadas en función de estos atributos

Un router BGP transmite y recoge rutas BGP a través de sesiones BGP establecidas con otros routers BGP. En función de la evolución de la red (averías, modificación de configuración, mantenimiento) los routers BGP intercambian mensajes, los cuales realizan las siguientes funciones:

- Informar a sus vecinos de las rutas nuevas (mensaje de withdrawn), o modificadas (mensaje de update).
- Adaptar su propia tabla de rutas en función de los mensajes que reciben.

Así pues, todo cambio estructural de la red puede generar mensajes BGP y en consecuencia una nueva fase de convergencia del protocolo BGP.

Sesiones iBGP

Es importante que dediquemos un apartado específico a estas sesiones porque serán muy importantes durante el desarrollo del proyecto.

Dos routers BGP intercambian su información a través de una sesión BGP. La sesión sólo puede establecerse si los dos routers conocen como enrutarse. En consecuencia, se distinguirán dos tipos de sesiones BGP (véase figura 9):

- Las sesiones iBGP, establecidas entre dos routers de un mismo AS.
- Las sesiones eBGP, establecidas entre dos routers de AS diferentes.

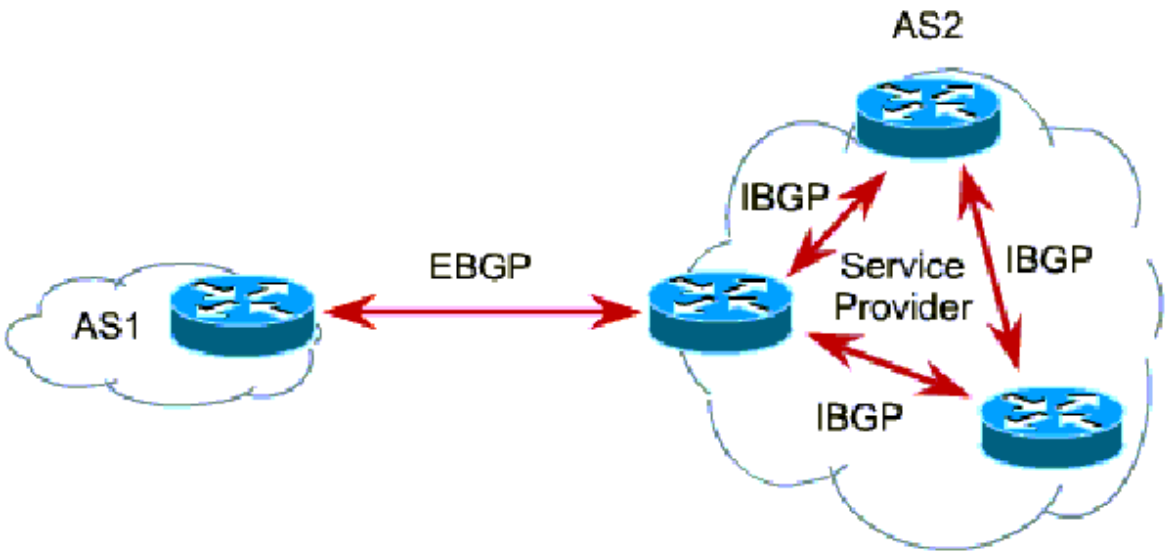


Figura 9: iBGP vs eBGP

Llamamos router de borde a un router que establece al menos una sesión eBGP. Por la política del protocolo, cuando un router de borde transmite por eBGP una ruta hacia un AS vecino, "aplasta" el antiguo nexthop BGP con una de sus propias direcciones. Esta dirección pasa a ser enrutable por el router al cual transmite la ruta mediante eBGP.

Las sesiones BGP se basan en el protocolo TCP, que garantiza un transporte sin error de los mensajes entre los dos routers.

Existen dos tipos de sesiones BGP, las establecidas entre routers adyacentes (atendiendo a IGP) que son llamadas sesiones monohop y las que se establecen entre routers no adyacentes, las sesiones multihop.

¿Cómo decide BGP?

En la actualidad, un router almacena alrededor de 250.000 rutas BGP en su tabla de ruta. Sin embargo un router BGP puede aprender varias rutas BGP con destino a un mismo prefijo. Se hablará entonces de que estas rutas son competidoras. Cuando un router tenga rutas competidoras deberá determinar cual utilizar.

El proceso de decisión BGP consiste en comparar en un determinado orden una serie de atributos BGP con el fin de elegir la ruta que debe utilizarse, llamada mejor ruta. Cada etapa de comparación permite estrechar el conjunto de rutas competidoras. Cuando sólo queda una, el router la determina como su mejor ruta y la instala en su tabla de rutas.

Vamos ahora a enumerar el proceso de decisión BGP y los atributos BGP comparados. La puesta en práctica del proceso de decisión puede ser diferente en función del fabricante, el que exponemos a continuación es el de la empresa Juniper Networks **[14].** Se evalúa el punto 1, en caso de empate pasamos al 2, y así sucesivamente.

1. El atributo preferencia local (*local-pref*) es un número positivo asociado a un nexthop BGP. Se prefieren las rutas en las que el atributo es máximo.

2. Se prefieren las rutas de *AS path* de longitud mínima: el atributo AS path lista los ASN de los AS cruzados por la ruta BGP. Este atributo permite introducir un concepto de distancia entre dominios. Un camino entre dominios será a priori más corto si cruza pocos AS. En la práctica, los AS pueden ser de tamaños muy diferentes, por lo tanto evaluar la longitud de un camino sobre el número de saltos inter-AS no es muy consistente.

3. Se prefiere la ruta de mejor origen: se prefiere una ruta aprendida por IGP sobre las rutas aprendidas por EGP, estas últimas se prefieren a las rutas cuyo origen no se especifica.

4. Una ruta aprendida por eBGP prevalecerá sobre las rutas aprendidas por iBGP. Recordemos que se despliega a priori una sesión eBGP entre dos router de borde que pertenecen a dos AS diferentes. Cuando un router de borde elige su mejor ruta según este criterio, eso significa que el router al cual se interconecta está en condiciones de transportar su tráfico hasta el prefijo destino. Así pues, esta etapa de decisión aplica una regla conocida como hot-potato routing: se prefiere enviar el tráfico hacia un AS vecino antes que de conservarlo en su propio AS.

5. Se prefieren las rutas BGP cuyo coste IGP para alcanzar el nexthop BGP es mínimo. Se busca el punto de salida lo más cerca posible (en el sentido de los pesos IGP). Este paso continua con el hot-potato routing, favoreciendo que los paquetes salgan cuanto antes del AS.

Si varias rutas competidoras siguen empatadas en este punto, pueden considerarse casi equivalentes. El router puede almacenar estas distintas rutas en su tabla de ruta (opción multipath). En ese caso puede distribuir su tráfico equitativamente hacia estos distintos puntos de salida. No obstante, sólo propaga a sus vecinos la mejor ruta, que deberá elegir de alguna manera, como puede ser atendiendo a otros atributos, por ejemplo cluster-list (eligiendo el más corto) o simplemente escogerla de forma aleatoria. Este proceso de desempate lo encontramos en la literatura con el nombre de tie-break.

En este proyecto fin de carrera hemos supuesto que las decisiones de BGP se realizan basándose en los puntos 4 y 5. Los pasos previos se toman utilizando atributos BGP, y nosotros hemos decidido no utilizar estos atributos en nuestras implementaciones al igual que se ha hecho en la literatura estudiada.

Por último, como comentamos anteriormente, este proceso puede ser diferente en otros fabricantes. Por ejemplo en la figura 10 se muestra el proceso de decisión que tomarán los routers de CISCO.

Order	Preference	Description
0. Synchronized	TRUE	Use only routes that meet the synchronization requirement
1. Weight	Highest	Administrative override
2. Local Preference	Highest	Used internally to pick path out of AS
3. Self originated	TRUE	Used to prefer paths originated on this router
4. AS-Path	Shortest	Minimize AS-hops
5. Origin	i<?	Prefer stability
6. MED	Lowest	Used external to come in
7. External	EBGP<IBGP	External path preferred over internal path
8. IGP cost	Lowest	Look for more information
9. EBGP Peering	Oldest	Prefer stability
10. RID	Lowest	Choose one with lowest BGP router ID

Figura 10: Orden de decisión de los routers BGP de CISCO

2.5 Topología full mesh en BGP

Las sesiones iBGP establecen una nueva topología o grafo sobre la red del AS. En este grado dos nodos están conectados si existe una sesión iBGP entre ellos. La topología más frecuente en BGP es el full mesh que se utiliza generalmente en los pequeños y medios AS, es decir, la gran mayoría de los AS que constituyen Internet. En un full mesh, cada router del AS establece una sesión iBGP hacia cada uno de los otros routers BGP del AS (véase figura 11).

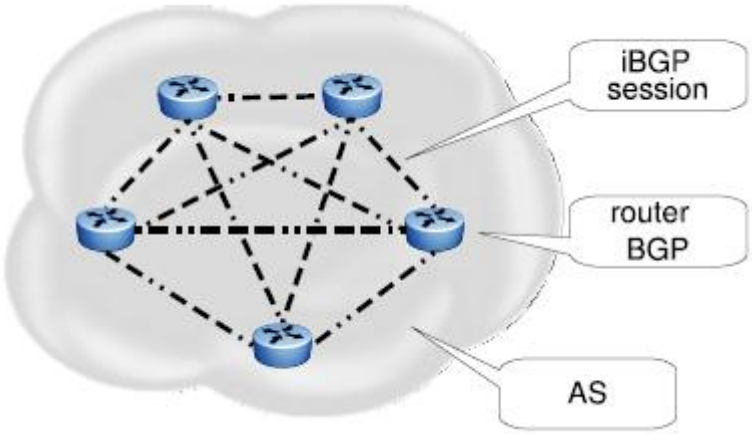


Figura 11: Full mesh BGP

Cada router tiene entonces conocimiento de las mejores rutas de todos los routers de su AS. Gracias a esta propiedad, el full mesh presenta las siguientes ventajas:

- Simplicidad. El comportamiento del full mesh es especialmente simple de prever.

- Diversidad máxima. Ninguna ruta se pierde durante la difusión iBGP. Cada router tiene conocimiento de la mejor ruta de cada router de borde. Así es más probable que cada router tenga una o más rutas auxiliares y en consecuencia la convergencia será más rápida.

- Convergencia rápida. Además de por la diversidad, la fase de convergencia se realizará en menos tiempo, ya que la velocidad de propagación de un mensaje BGP depende directamente del número de routers BGP que la evalúan. En un full mesh, se transmite inmediatamente un mensaje de un router de borde a todos los routers del AS.

- Robustez. Cualquiera que sea el router BGP que cae averiado, se propaga un mensaje BGP recibido por todos los routers BGP. Así en caso de avería, cada router del AS dispone rápidamente de la información necesaria para poder utilizar una ruta auxiliar.

Desgraciadamente, si un AS contiene un número muy elevado de routers BGP, la solución de full mesh iBGP no escala bien debido a las siguientes limitaciones:

1. Tamaño de las tablas de ruta: Un router propaga hacia sus vecinos iBGP la totalidad de sus rutas aprendidas en eBGP. En términos de memoria la instalación de sesiones iBGP resulta costosa por lo que es conveniente no establecer demasiadas.

2. Topología poco práctica. Si incluimos o eliminamos un router en el AS es necesario reconfigurar todos los routers BGP del AS. Si el número de routers presentes en el AS es grande, esta manipulación puede resultar especialmente tediosa y conducir a errores de configuración (errores humanos), en caso de que sea necesario establecer rutas de forma manual.

3. Gran cantidad de mensajes BGP transmitidos. Un full mesh es especialmente sensible al menor acontecimiento que ocurre en la red. En cuanto que un router realice un cambio en una ruta, enviará un mensaje de actualización a todos sus vecinos, incluidos los que no se ven afectados por esta modificación. Esto genera entonces una actividad inútil.

Actualmente existen dos líneas de trabajo que permiten solucionar las limitaciones del full mesh iBGP:

- *Las confederaciones BGP.*
- *La reflexión de rutas.*

Ambas se comparan en Dube 1999 **[3]** y en Scudder y Dube 1999 **[4].** A continuación describimos someramente cada una de ellas.

2.6 Confederaciones BGP

Cuando el full mesh iBGP no es técnicamente posible, una primera solución consiste en utilizar confederaciones BGP **[15]**. La idea básica consiste en subdividir un AS (demasiado grande para utilizar full mesh) en pequeños sub-AS llamados confederaciones BGP. Se asocia a cada uno de estos sub-AS un ASN privado, que sólo tiene validez en el AS al cual pertenecen. Estos ASN privados no se exportarán en las sesiones eBGP inter-AS, volviendo así la subdivisión en confederaciones completamente invisible fuera del AS. Además, los ASN que describen a las confederaciones no intervienen en el proceso de decisión BGP.

El concepto de confederación tiene un impacto directo en términos de decisión de transporte BGP, el coste IGP se evaluará hacia el nexthop BGP de la confederación a la que el router pertenece. De esta forma, sólo los costes IGP de los enlaces internos a la confederación afectan al transporte BGP de este sub-AS. Cada confederación utiliza su propia topología IGP, e ignora las de las otras confederaciones. Por lo tanto, es más delicado enrutar el tráfico hacia fuera del AS.

Aunque resuelve el problema de la escalabilidad, esta solución presenta otros inconvenientes si la comparamos con el full mesh iBGP:

- Pérdida de flexibilidad y suboptimización. A priori se despliega un IGP por confederación, por lo que, no es posible cuantificar la distancia que separa un router de los diferentes puntos de salida. Además el mejor punto de salida de la confederación no se sitúa siempre sobre el camino que conduce al mejor punto de salida del AS. En ese caso, el tráfico permanece mucho más tiempo del necesario dentro del AS. Esto genera una sobrecarga inútil de la red y puede alargar el tiempo de transporte de los paquetes.

- Complejidad. El transporte se vuelve más complejo de comprender. Al contrario que en full mesh, el punto de salida utilizado no es siempre el más cercano posible.

- Convergencia. El transporte BGP converge más lentamente, ya que un anuncio de ruta es tratado por más routers BGP que en el caso de un full mesh.

- Redundancia. Una misma ruta BGP puede ser aprendida por caminos de difusión diferentes en el AS, y en consecuencia tratarse varias veces por un mismo router.

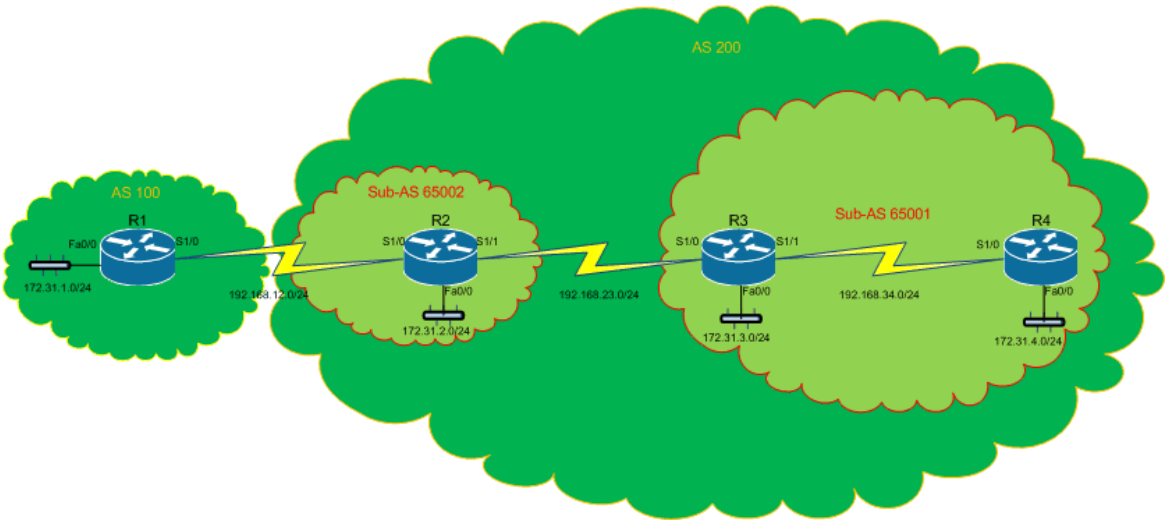


Figura 12: Confederaciones BGP

¿Una confederación por router?

El reparto en confederaciones nos obliga a no utilizar una topología IGP global a todo el AS. En efecto, cada confederación puede utilizar su propio IGP, y no ver más que la topología de la red que engloba. Llevando esto al extremo podríamos imaginar que utilizamos a una confederación por cada router BGP y así prescindir de IGP. De esta forma el tráfico interno a un AS se enruta también gracias a BGP. Este enfoque tiene dos limitaciones:

1. El protocolo BGP converge mucho más lentamente que un protocolo a estado de enlace, con lo que la red será menos reactiva a las averías internas.

2. Los ASN privados no intervienen en el proceso de decisión BGP cuando se comparan caminos de AS. Los enlaces inter-confederación no pueden ser ponderados por costes IGP, ni por AS path prepending. Es complejo por lo tanto pedir puntos de salida según criterios de hot potato routing.

2.7 Reflexión de rutas

La reflexión de rutas permite desplegar BGP en un AS de tamaño importante, cuando el full mesh no es posible. Podemos encontrar una amplia descripción de esta solución al problema de escalabilidad del full mesh iBGP en Bates 2006 **[7]**. Esta técnica consiste básicamente en autorizar a algunos routers, llamados RRs (Route Reflectors), a propagar algunos mensajes aprendidos en iBGP a vecinos iBGP. Al romper esta norma de propagación, ya no es necesario establecer una sesión iBGP entre cada par de routers BGP del AS. Esta solución se utiliza comúnmente en los AS de gran tamaño.

En principio un router BGP puede tener tres tipos de vecinos iBGP:

1. Routers clientes iBGP (eso significa que es RR de estos routers). (Sesión UP)
2. Routers vecinos iBGP "clásicos". (Sesión OVER)
3. Routers de los cuales es cliente. (Sesión DOWN)

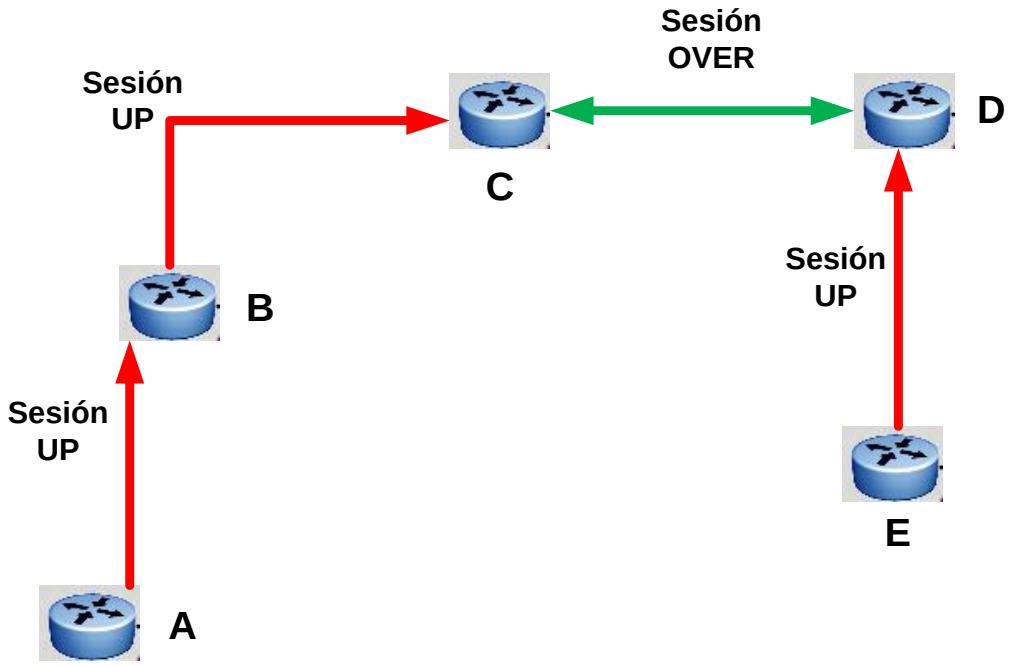


Figura 13 : Tipos de sesiones BGP

En la figura 13 podemos observar los tres tipos de vecinos mencionados, por ejemplo, C y D son vecinos iBGP "clásicos" y "A" es cliente de "B" que a su vez es cliente de "C".

En la reflexión de rutas se establece que ***un RR puede propagar todo mensaje iBGP hacia sus clientes iBGP***. Además de esto, ***puede transmitir todo mensaje procedente de***

un cliente iBGP hacia el conjunto de sus vecinos iBGP. Como siempre en BGP, **un RR propaga para cada destino sólo su mejor ruta a sus vecinos BGP.**

La topología de RRs es generalmente jerárquica. Por ejemplo, los routers nacionales son clientes de routers continentales, ellos mismos clientes de routers intercontinentales, etc (figura 13). Según el tamaño de la red una topología de RR puede implicar varias jerarquías de RRs (figura 14).

Llamamos cluster al conjunto de routers formado por un RR y sus clientes. Cada cluster está definido por un número llamado *cluster-id*. El cluster-id se configura en cada RR. Cada vez que un RR recibe un mensaje BGP, añade su cluster-id al final del *cluster-list* (el cluster-list es otro atributo al igual que el AS path, aplica una forma de path vectoring). Un mensaje BGP transporta pues la secuencia de RRs que ha cruzado. Si un RR se entera de una ruta por iBGP y detecta su cluster-id en el cluster-list, la ignora. Este mecanismo permite evitar que un mensaje se propague indefinidamente a lo largo de un ciclo de RRs. Por esto, el cluster-list frena tal propagación de modo que el mensaje sólo cruce a lo sumo una vez cada RR. El cluster-list no se exporta fuera del AS y vuelve la reflexión de rutas completamente invisible para los otros AS.

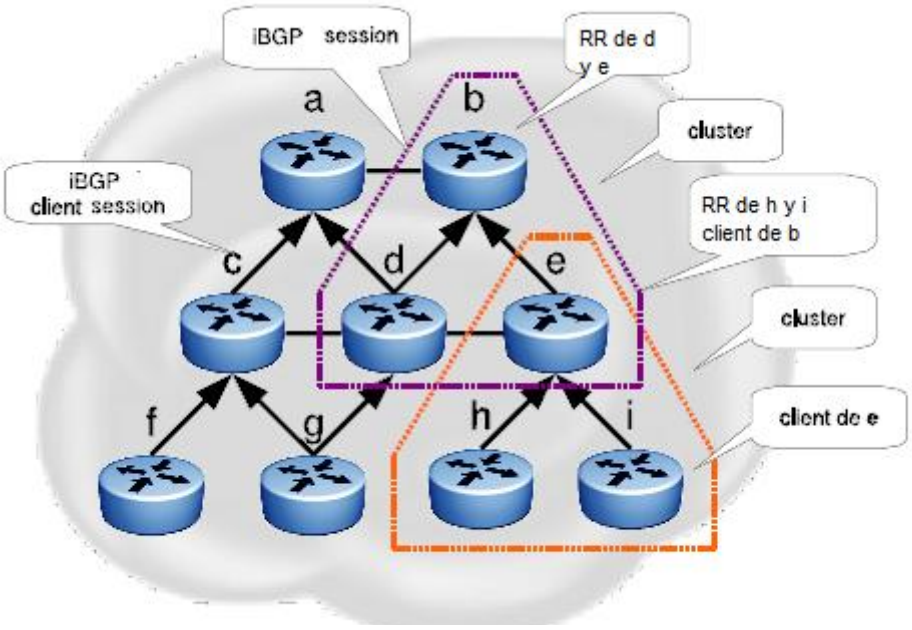


Figura 14: Cluster y jerarquía de RRs

Al contrario que con las confederaciones BGP, podemos basarnos en la topología IGP total del AS para enrutar el tráfico hacia el punto de salida más cercano posible. Pese a todo existen problemas se presentan con todo detalle en el apartado 2.9.

2.8 Topologías usadas en la reflexión de rutas

Como hemos mencionado anteriormente, la mayoría de las topologías de reflexión de rutas utilizadas en la actualidad son jerárquicas. Un RR puede ser cliente de otro RR más importante y así sucesivamente. La elección del número de RRs es determinante y debe elegirse con cuidado, ya que cuantos más RRs, más se mejoraran las decisiones a la hora de enrutar, pero también más mensajes serán intercambiados. Por estas razones, es necesario encontrar un compromiso entre el número de RRs y el número de rutas que cada cliente debe conocer para alcanzar un destino (cuanto menor sea el número de RRs, menos mensajes se intercambiaran).

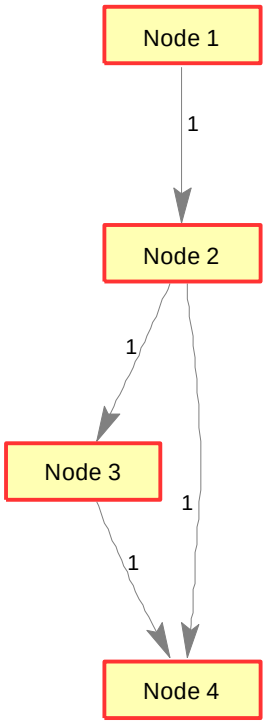


Figura 15: Ejemplo jerarquía de RRs

La figura 15 está extraída directamente de una de nuestras pruebas. Muestra un ejemplo muy sencillo de jerarquía de reflectores, que solo dispone de 4 routers. El nodo 4 será el reflector de mayor jerarquía y el nodo 2 el de menor (el nodo 1 sólo es cliente, no es reflector).

Dejando de lado el inciso de la imagen, hemos de destacar que aunque parece interesante establecer el mínimo número de sesiones iBGP, es también necesario incluir

cierta redundancia para hacer el sistema más robusto, esta redundancia se conoce en la literatura como diversidad de rutas, Uhlig y Tandel 2009 **[8]**. La redundancia genera un número de sesiones iBGP suplementarias y por consiguiente el numero de mensajes intercambiados también aumenta, con lo que esta redundancia tiene un impacto directo sobre la carga de los routers.

Una de las reglas que hemos encontrado en la literatura y que por tanto hemos seguido en nuestra programación, es que es preferible establecer sesiones iBGP monohops antes que multihops. Esto se debe básicamente a dos razones:

- La primera es que se impide la aparición de algunos cierres de transporte (Rosen et al. 1999 **[17]**.) causados por ciclos. Un cierre de transporte implica la pérdida de la información transmitida, debido a que los paquetes que la transportan se descartan.

- La segunda se sustenta en que es más fácil comprender una red cuya topología iBGP está de acuerdo con la topología IGP de la red. Por esta razón, en el programa que implementamos para diseñar topologías BGP, tuvimos en cuenta el peso IGP a la hora de establecer la sesiones iBGP.

Para poner un ejemplo sencillo de topología de reflexión de rutas válida podemos mencionar la llamada quasi-full mesh. La topología quasi-full mesh es una topología en la que los reflectores de ruta están unidos entre sí mediante full mesh, y todos los demás router (los clientes) tienen sesión directa con cada reflector. La condición necesaria para ser un reflector es ser un router de borde, es decir tener instalada alguna sesión eBGP.

Esta topología sólo es interesante en redes en las que el número de routers de borde en relación con el número total de routers es bajo. En caso contrario la reducción del número de sesiones con respecto a full mesh sería mínima. Como ejemplo vamos a imaginar una red de 20 routers en la que únicamente tenemos 4 routers de borde. En esta situación, con la topología quasi-full mesh tendríamos full mesh en estos 4 routers (4 x 3 = 12, son 6 sesiones de tipo OVER, que deben contarse dos veces dada la naturaleza de esta sesión) y enlaces directos desde cada uno de los restantes routers (clientes) hasta los reflectores ((20-4) x 4 = 64 sesiones de tipo UP), por lo que en total tendríamos 76 sesiones iBGP instaladas. Un full mesh en una red de 20 nodos precisa de 380 sesiones (19 x 20) por lo

que conseguiríamos una reducción del 80 % en el número de sesiones, (380 / 76 = 5). A priori puede parece interesante utilizar esta topología, pero en las redes reales, el número de routers de borde es bastante grande en relación con el número de routers total, por lo que no trabajaremos con ella. Por ejemplo, en la red tier-1 que se utiliza en la sección 4.7, casi todos los routers son routers de borde por lo que no obtendríamos una reducción en el número de sesiones apreciable con respecto al full mesh. La figura 16 muestra un ejemplo de una red de 6 nodos en las que 4 son reflectores.

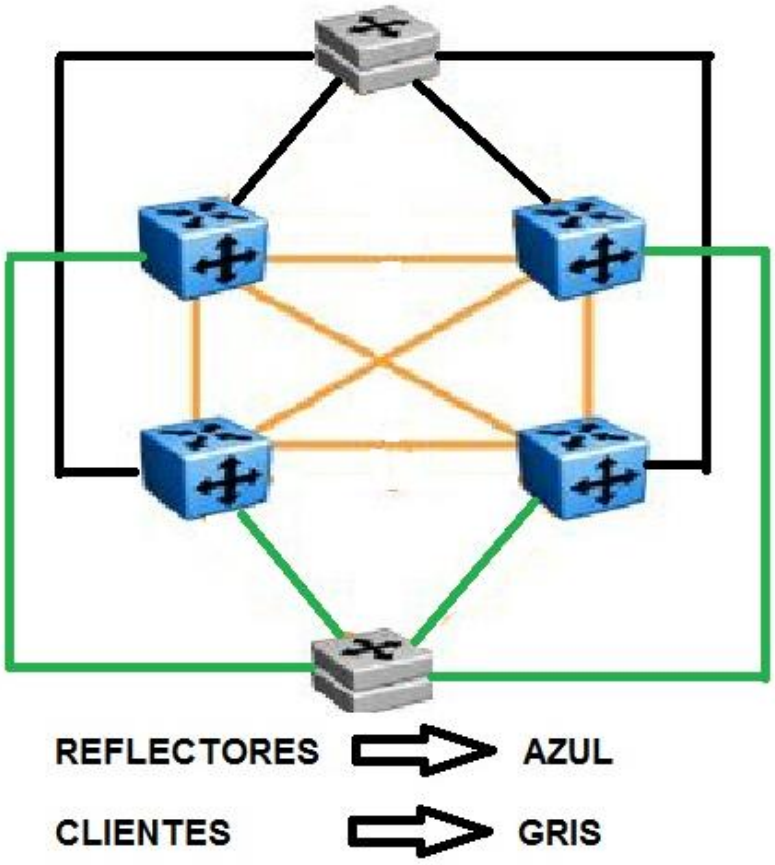


Figura 16: Topología quasi-full mesh

Como se indica en la imagen, los routers en azul son reflectores que como vemos están totalmente mallados entre sí. Los routers grises son clientes porque no tienen sesiones eBGP. En este caso el número de sesiones es 20 [([Líneas naranjas (6)] x 2) + (Líneas verdes [4]) + (Líneas negras [4])]. En un full mesh tendríamos 6 x 5 =30 sesiones (recordar que la sesiones OVER han de contarse dos veces), por lo que nos ahorraríamos un tercio de las sesiones.

2.9 Problemas encontrados al utilizar la reflexión de rutas

En este apartado presentamos los problemas que se producen en un AS que utiliza reflexión de rutas BGP. Cuando se elige mal una topología de reflexión rutas, el enrutado BGP puede ser:

- Subóptimo (véase sección 2.9.1)
- Víctima de ciclos de transporte y cierres de transporte (véase sección 2.9.2),
- No determinista (véase sección 2.9.3),
- Inestable (véase sección 2.9.4),

Vamos a ilustrar estos problemas sobre topologías de red muy simples. Para facilitar su comprensión, mostraremos sobre un mismo grafo las topologías IGP e iBGP.

Los grafos utilizados seguirán el siguiente formato:

- Nodos:
  - Su nombre viene escrito en rojo.
  - Si es nexthop (gateway) o RR se indicará en el nombre

- Arcos:
  - Representamos una sesión eBGP o iBGP por un arco en discontinua. Con el fin de reducir el grafo, sólo las sesiones UP están representadas (las sesiones DOWN simétricas se omiten) para simbolizar una sesión iBGP establecido entre un cliente y un RR. Representamos las sesiones OVER con una doble flecha.
  - Se representa un enlace IGP por un arco en línea continua. Su peso (métrica) estará escrito en negro

2.9.1 Sub-optimalidad

La figura 17 pone de manifiesto que el enrutado BGP no es siempre óptimo cuando se elige mal la topología de reflexión de rutas. Recordemos que un enrutado subóptimo genera una utilización subóptima de los recursos de la red y el tiempo de transporte es a priori más largo, ya que el tráfico utiliza un camino más largo. Es necesario recordar que la elección de la mejor ruta de un RR depende generalmente de su posición en la topología IGP. La ruta que el RR juzga como mejor no es necesariamente la más interesante desde el punto de vista de los routers hacia los cuales va a propagar la ruta (clientes).

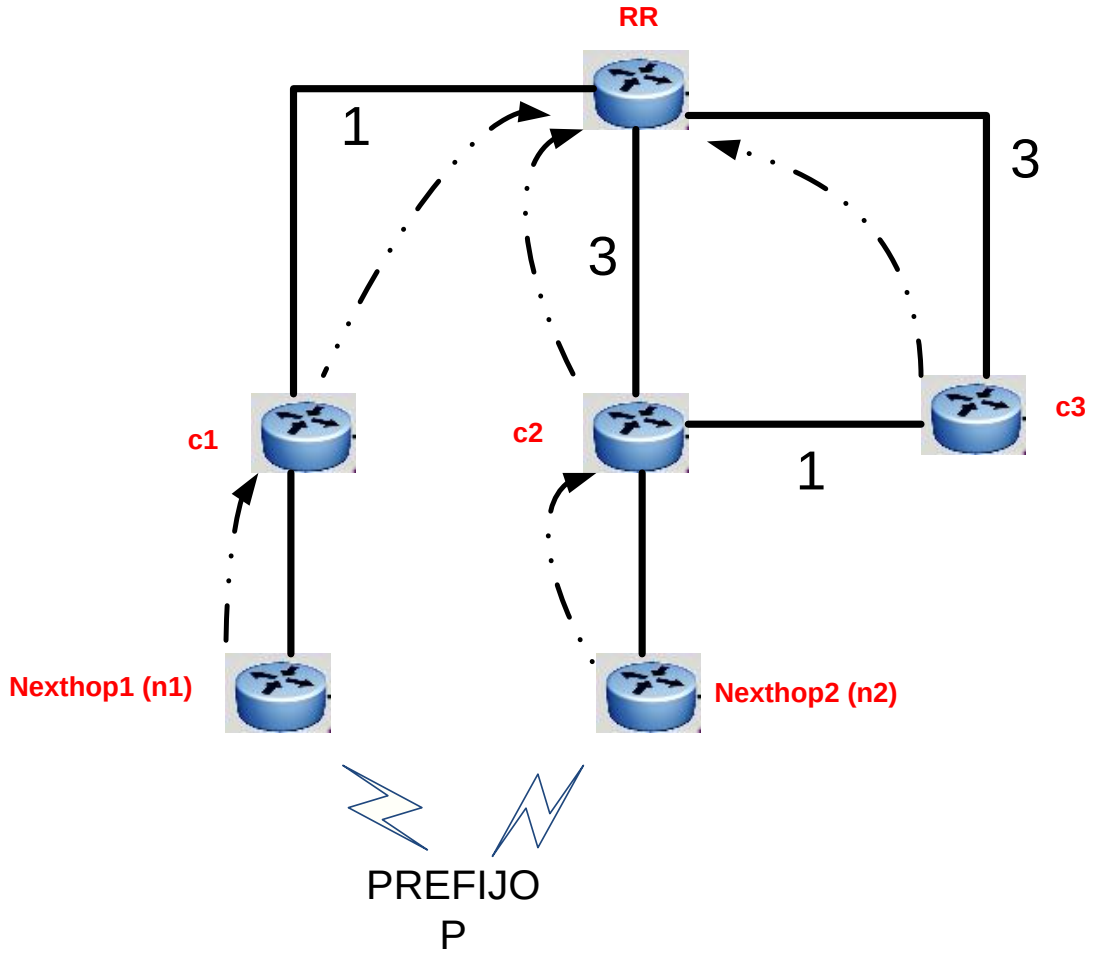


Figura 17: Ejemplo de sub-optimalidad

En la figura 17 se muestran dos routers de borde, *c1* y *c2*, que son competidores para un prefijo *P*. Eligen respectivamente las rutas anunciadas por *n1* y *n2*. Esta elección no se pondrá nunca en duda, ya que preferirán siempre una ruta aprendida por eBGP sobre una ruta aprendida por iBGP.

Ambos routers transmiten su mejor ruta hacia el reflector *RR*, que prefiere la ruta de *c1* (con coste IGP igual a 1), antes que la ruta de *c2* (coste IGP igual a 3).

Cada router propaga el tráfico con destino a *P* del siguiente modo:

- *c1* transmite el tráfico hacia *n1* (ruta aprendida por eBGP).
- *c2* transmite el tráfico hacia *n2* (ruta aprendida por eBGP).
- *RR* transmite el tráfico hacia *c1* (por el camino más corto de *RR* a *c1*, [*RR, c1*]).
- *c3* transmite el tráfico hacia *RR* (por el camino más corto de *c3* a *c1*: [*c3, RR, c1*]).

Se observa que *c3* pasa su tráfico hacia *c1* en vez de *c2*. Así el tráfico de *c3* sigue el camino [*c3, RR, c1*] (coste IGP igual a 4) en lugar de [*c3, c2*] (coste IGP igual a 1). Además del coste IGP suplementario, el tráfico ocupa tres router del AS en lugar de dos, y genera una sobrecarga inútil de los equipamientos de la red.

La solución más sencilla consiste en corregir el diseño iBGP. Este problema se debe a un mal diseño iBGP. En concreto, *RR* encubre la ruta anunciada por *n2* al router *c3*. Esto podría solucionarse en instalando a una sesión iBGP entre *c2* y *c3*.

2.9.2 Bucles y cierres de transporte

Al producirse un bucle en el camino que siguen los paquetes se provoca un cierre de transporte, lo cual causa que el paquete se pierda, por esto es importante evitar que se produzca.

El ejemplo de la figura 18 ilustra el caso de la aparición de un bucle de transporte que conduce a un cierre de transporte. Eso significa, como acabamos de comentar, que cualquier paquete capturado en el bucle no alcanzará nunca su destino.

Los nexthop *n1* y *n2* emiten cada uno una ruta competidora y casi equivalente con destino a un prefijo *P*. *RR1* y *RR2* eligen respectivamente las rutas de *n1* y *n2* y conservan esta elección ya que prefieren una ruta aprendida en eBGP sobre una ruta aprendida por iBGP. Ambos transmiten su ruta a su cliente respectivo. Así *c1* no puede elegir otra ruta que no sea la ruta anunciada por *n1* y lo mismo sucede con *c2* al que solo le llegan las rutas anunciadas por *n2*. De esta forma, ni *c1*, ni *c2* tienen conocimiento de su punto de salida óptimo atendiendo al coste IGP (respectivamente *n2* y *n1*).

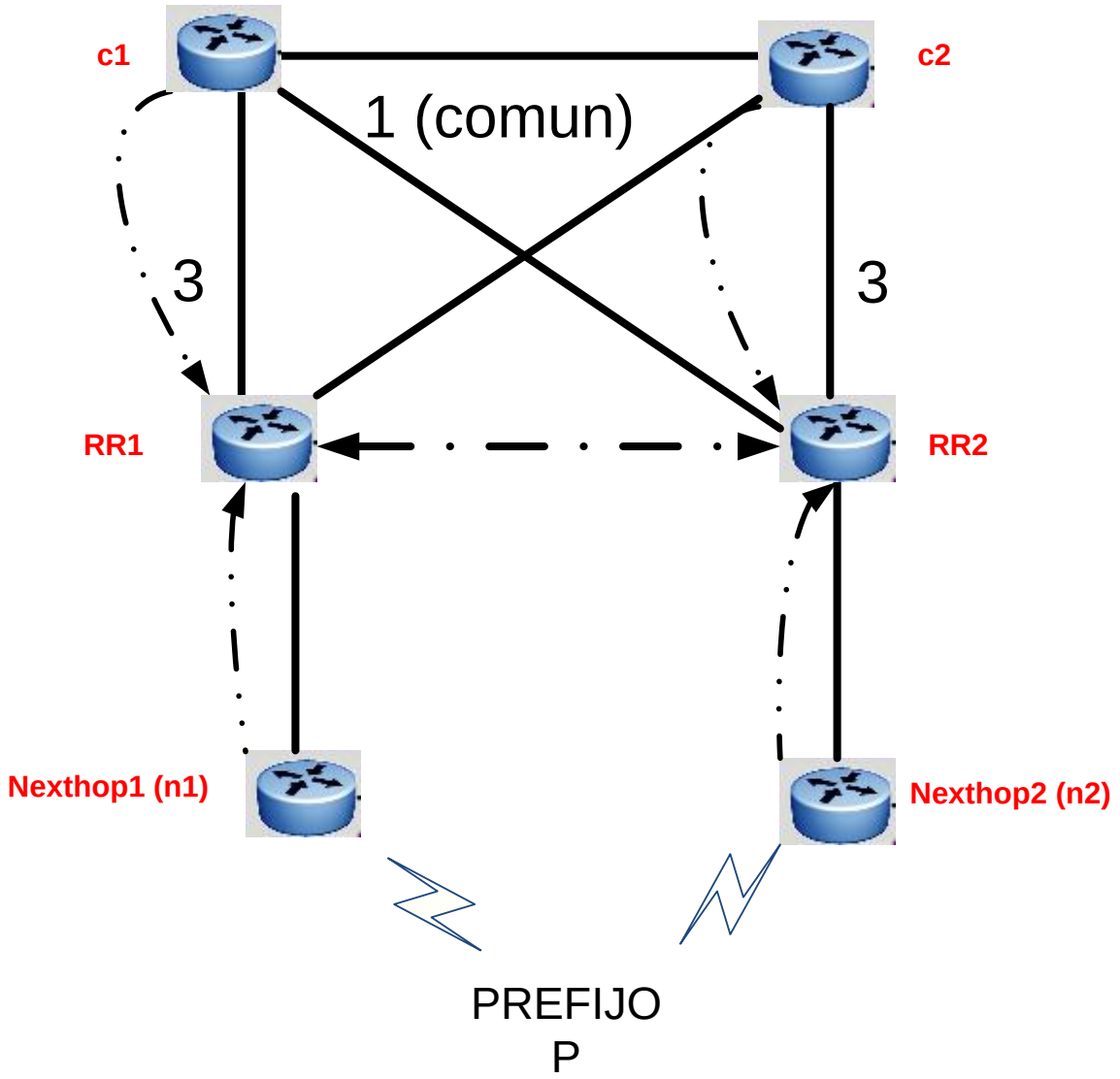


Figura 18: Ejemplo bucle→ cierre de transporte [1 común pertenece a los 3 enlaces IGP]

Cada router propaga el tráfico con destino a *P* del siguiente modo:

- *RR1* transmite el tráfico hacia *n1* (por el camino más corto de *rr1* a *n1*: [*RR1, n1*], eBGP).
- *RR2* transmite el tráfico hacia *n2* (por el camino más corto de *rr2* a *n2*: [*RR2, n2*], eBGP).
- *c1* transmite el tráfico hacia *c2* (por el camino más corto de *c1* a *n1*: [*c1, c2, RR1, n1*]),
- *c2* transmite el tráfico hacia *c1* (por el camino más corto de *c2* a *n2*: [*c2, c1, RR2, n2*]).

Podemos observar que si *c1* debe emitir tráfico hacia *p*, entonces éste transmite su tráfico hacia *c2* para poder de alcanzar *n1*, *c2* al recibir este tráfico, y observar que su destino es *P*, mira en su tabla de rutas y decide que para llegar a *p* debe salir por el nexthop *n2*, de esta forma devuelve el tráfico hacia *c1* (inicio del bucle). *c1* aplica entonces el mismo razonamiento y devuelve el tráfico a *c2* (bucle completo). Con estas decisiones, el tráfico se devuelve constantemente entre *c1* y *c2*, terminando así en un cierre de transporte.

Una solución posible a este conflicto consiste en la aplicación del protocolo MPLS, (Multiprotocol Label Switching) **[16],** que es un protocolo que se sustenta en el establecimiento de túneles. Cuando se autoriza a un tráfico a penetrar en un túnel MPLS, sigue este túnel sin preocuparse de las rutas de los routers cruzados. Una vez salido del túnel, el tráfico se enruta normalmente. Bastaría entonces con establecer un túnel MPLS de *c1* a *RR1* y *c2* a *RR2*.

Con esta solución lograríamos solucionar el problema del bucle que conduce al cierre de transporte. Sin embargo, el transporte seguiría siendo subóptimo en términos de coste IGP. Además de esto, este enfoque requiere la configuración de los túneles sobre los routers.

Si nos centramos en el tema que ocupa este proyecto, la búsqueda de topolgías iBGP óptimas, podemos encontrar una solución más sencilla, corregir el diseño iBGP:

c1 debería enrutar su tráfico con destino a *P* hacia *n2*, y *c2* hacia *n1*. Este transporte puede obtenerse instalando una sesión iBGP entre *c1* y *RR2*, y otra entre *c2* y *RR1*. Así *c1* y *c2* enrutarán respectivamente el tráfico con destino a *P* hacia *n2* y *n1*, mediante *RR2* y *RR1*. Como podemos ver, un mejor diseño iBGP sigue siendo la mejor solución para el problema.

2.9.3 No determinismo

El transporte de una red no debería nunca depender del orden en el cual los routers aprenden las rutas. La figura 19 ilustra un caso de transporte no determinista. Para entenderlo, vamos a estudiar el caso en el que *n1* anuncia su ruta antes que *n2*, y posteriormente el caso donde *n2* anuncia su ruta antes que *n1*.

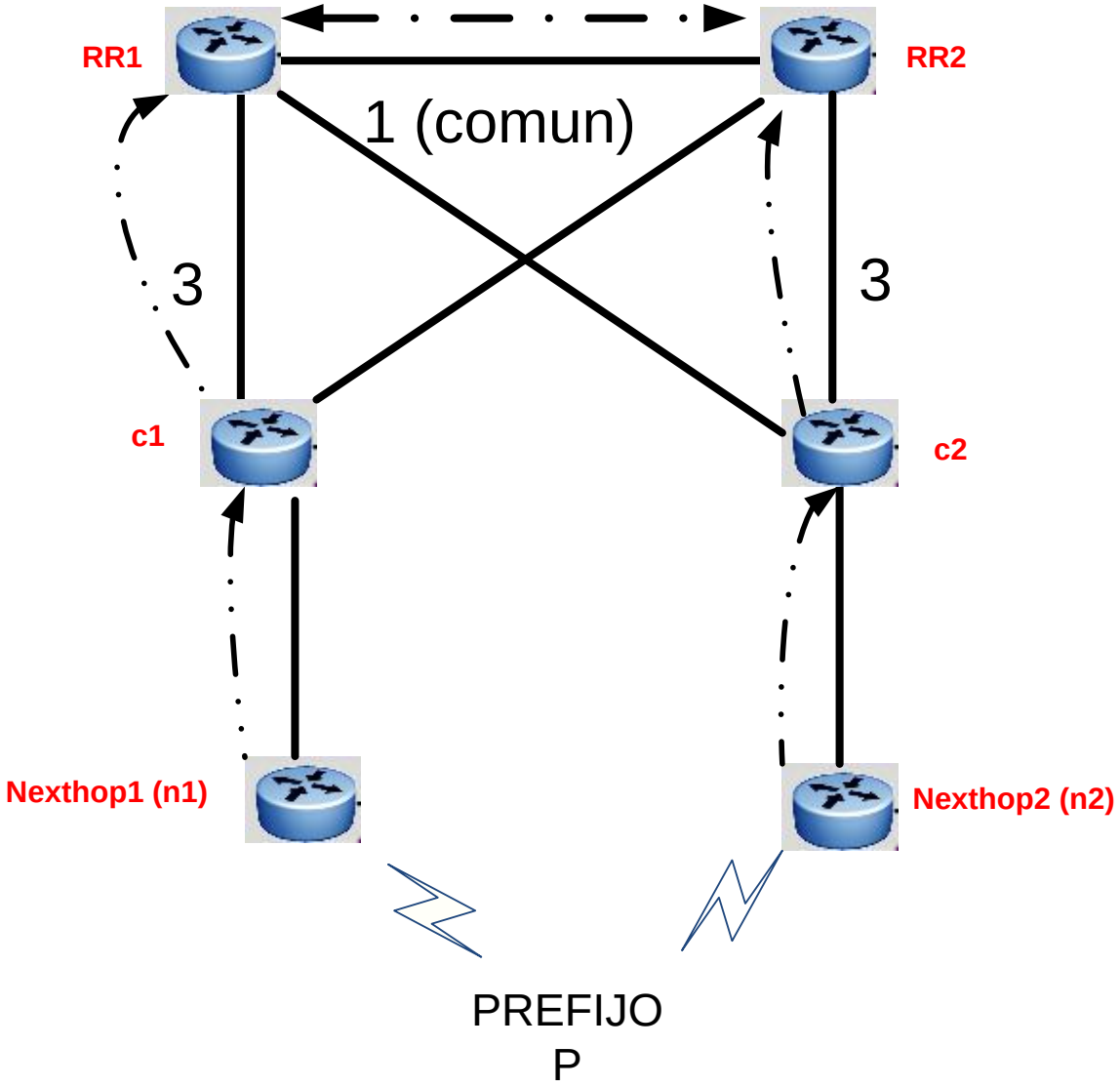


Figura 19: Ejemplo: caso no determinista

- Supongamos que la ruta anunciada por *n1* llega primero. Se propaga entonces en todo el AS (routers *c1, RR1, RR2, c2*). Supongamos ahora que *n2* anuncia una ruta competidora y casi equivalente a la ruta anunciada por *n1*. Entonces *c2* elige la ruta anunciada por *n2* ya que prefiere una ruta aprendida por eBGP sobre una ruta aprendida por iBGP. Transmite a continuación esta ruta a *RR2*, pero éste sigue prefiriendo la ruta anunciada por *n1*. Así la propagación de la ruta anunciada por *n2* permanece aislada en *c2*. Al final, *c1, RR1, RR2* eligen la ruta anunciada por *n1* y *c2* la ruta anunciada por *n2*.

Ahora veamos el otro caso, supongamos que la ruta anunciada por *n2* llega antes que anunciada por *n1*. Con el mismo razonamiento, *c2, RR1 y RR2* eligen la ruta anunciada por *n2* y *c1* la ruta anunciada por *n1*. Con esto demostramos que, según el orden de llegada de las rutas, *RR1* y *RR2* convergen hacia la ruta anunciada por *n1* o *n2*. Se dice entonces que los dos routers, *RR1* y *RR2*, convergen hacia un transporte no determinista.

2.9.4 Inestabilidad

De este problema solo vamos a mencionar que su resultado es muy similar al explicado en el apartado anterior, el no determinismo. La principal diferencia es la causa que lo produce, si en el caso anterior era el orden de llegada de las rutas, en este caso será que los atributos de estas rutas cambien.

Este problema no lo tendremos en cuenta ya que hemos restringido estos cambios en las rutas, con lo que al no producirse este problema no podrá ocurrir.

2.10 ¿Cuál es el objetivo de una topología de RRs?

Hemos comentado en varias ocasiones que un mal diseño de reflexión de rutas podía conducir a una mala propagación de las rutas BGP en un AS. Si esto se produce, los routers no tienen toda la información necesaria para calcular un transporte coherente y eficaz. En este tipo de situaciones, el enrutado puede ser inestable, subóptimo o no determinista. Estos problemas son a menudo complejos de detectar, y además, es difícil garantizar que no vayamos a acabar en esa situación tras una avería.

El full mesh iBGP da solución a estos problemas, pero sólo es utilizable si el número de routers BGP presentes en el AS es pequeño. Entonces es el momento de preguntarnos, ¿es posible unificar las ventajas del full mesh y de la reflexión de rutas? Y en caso de ser posible, ¿será además técnicamente posible?

Una de las primeras soluciones que encontramos en la literatura es la propuesta en Vutukuru et al. 2006 **[9]**. Como solución al problema proponen que se establezca una sesión iBGP entre cada par de routers de borde (los que nosotros llamamos nexthop en nuestra programación) y que cada uno de ellos sea reflector de los demás routers del AS (anteriormente mencionada como topología quasi full mesh). Esta propuesta nos reduce el número de sesiones iBGP que tenemos instaladas en el AS, pero en la actualidad y más concretamente en las redes que estamos estudiando, la gran mayoría de routers actúan como routers de borde por lo que la reducción del número de sesiones instaladas no será muy significativa.

También en Anuj Rawat and Mark A. Shayman, 2006 **[19]** en el que sus autores además de presentar los problemas que se producen al utilizar reflexión de rutas, introducen un enfoque de diseño topologías. Los autores buscan minimizar la distancia IGP entre dos vecinos iBGP, este aspecto será clave en nuestra programación. Parece lógico seguir esta "norma" a la hora de construir topologías BGP ya que dos routers cercanos en el sentido IGP harán elecciones de enrutamiento BGP similares y además de esto, es más probable que nuestras sesiones sean monohop, o en el caso de ser multihop, tenderán a cruzar pocos routers debido a la penalización por la distancia. Con esto el encaminamiento o transporte BGP será más eficaz.

Ya en 2007 en Breitbart et al. 2007 **[2]** se comienza a hablar de diseñar algoritmos para la creación de topologías BGP que funcionen correctamente. Estudian entre otros aspectos como elegir los reflectores para minimizar el coste total IGP asociado a las sesiones iBGP que se instalen. Esta máxima ha sido tenida en cuenta por otros autores (nosotros también) a la hora de implementar algoritmos capaces de construir topologías de reflexión de rutas. En el mismo año Buob et al. 2007 **[6]** introducen el concepto de fm-optimalidad: una red es fm-óptima si la propagación de sus rutas se realiza de la misma manera que lo haría con una topología full mesh. En estas condiciones (red fm-óptima), aseguran que todos los routers reciben la información necesaria para poder enrutar cualquier prefijo y que ***si una topología es fm-óptima no sufrirá ninguno de los problemas típicos de la reflexión de ruta expuestos en el apartado anterior.*** Para estudiar la propagación de las rutas utilizan el "grafo extendido", concepto que nosotros también utilizamos. Podemos encontrar una descripción del mismo en el apartado 3.1.1.3.

Los mismos autores pero ya en 2008 (**[1]**), proponen un algoritmo para construir topologías fm-óptimas a partir de la red IGP que sigue la máxima expuesta en Breitbart et al. 2007 **[2]**. Este algoritmo es el punto de partida de nuestros desarrollos. Puede encontrarse su descripción en el apartado 3.2.1.

3

Diseño y Desarrollo

3.1 Consideraciones previas

Para poder entender tanto la programación como nuestros resultados vamos a introducir los conceptos básicos que hemos utilizado durante la realización del proyecto, que han sido:

- Grafos (igp, bgp, grafo extendido).
- Matrices (Representación matricial de estos grafos).

3.1.1 Modelos de grafos utilizados

Se describe la red de un AS con ayuda de dos grafos (el grafo IGP y el grafo BGP). Estos modelos de grafos se emplean generalmente en la literatura. En los grafos cada router estará representado por un nodo.

3.1.1.1 El grafo IGP

Se representa cada router de un AS por un nodo. Cuando un enlace IGP se instala de un router *u* a un router *v*, se construye un arco orientado *(u, v)* ponderado por el peso IGP correspondiente.

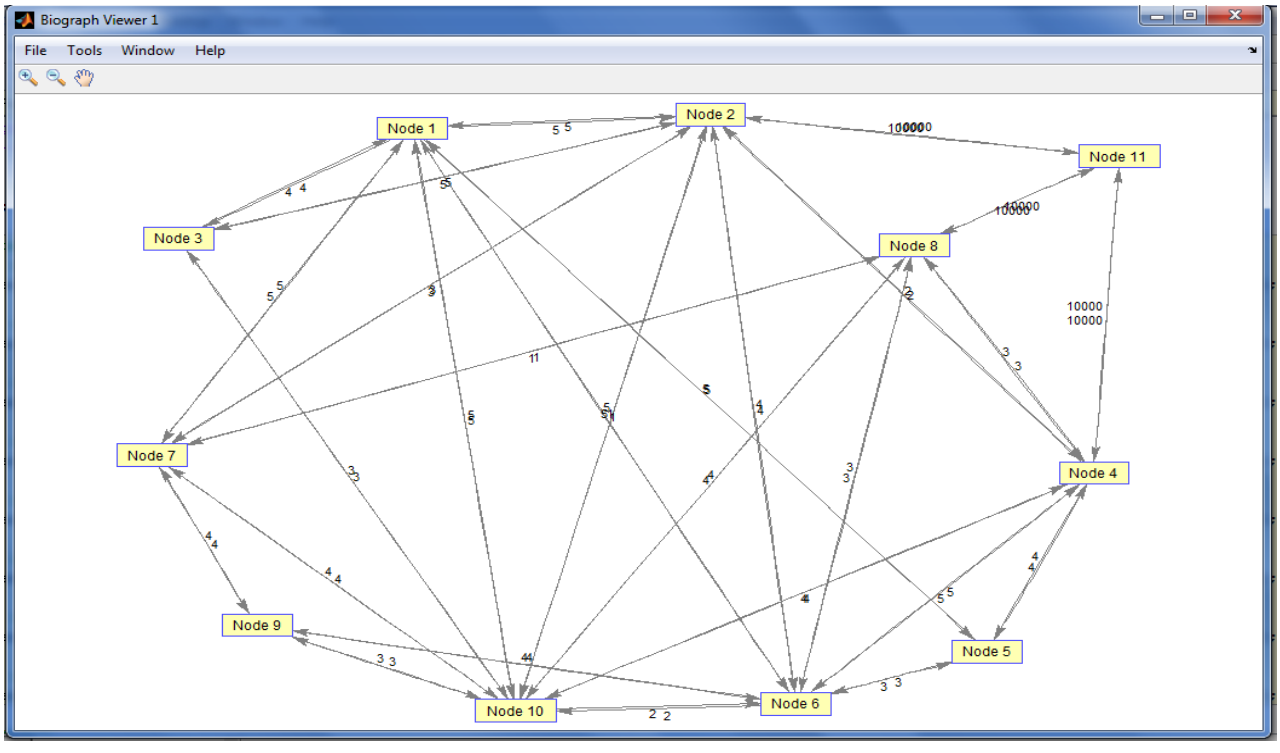


Figura 20: Grafo IGP extraído de MATLAB

La figura 20 muestra un grafo IGP creado a partir de los datos de una red IGP mediante MATLAB, en ella se muestra una de las redes iniciales que usaremos en nuestras primeras pruebas, cada nodo se corresponde con un router. Como podemos ver, los enlaces entre ellos son ponderados por un número que simboliza el peso IGP. La red mostrada en la figura 20 es una red muy mallada en sentido IGP, es prácticamente una red full mesh IGP, esto no es lo habitual en las redes reales donde el número de conexiones IGP suele ser mucho menor.

Es necesario distinguir el arco *(u, v)* y el arco *(v, u)* que no tienen por qué tener el mismo peso IGP. Denominamos *Gigp = (Vigp, Eigp)* al grafo orientado que representa la red IGP. Vigp serán los nodos, Eigp los enlaces.

Llamamos $|u , v|$ a la longitud más corta del camino de *u* a *v*. Esta distancia puede calcularse fácilmente con el grafo Gigp con ayuda de algoritmos de la teoría de grafos como por ejemplo el algoritmo de Dijkstra.

Si una avería *p* ocurre, se tiene en cuenta $|u , v|p$, la longitud del camino más corto de *u* a *v* en caso de que *p* suceda. Observemos que el camino utilizado en una avería *p* es al menos tan largo como el camino sin avería, con lo que siempre $|u , v| \leq |u , v|p$.

Diremos que dos routers *u* y *v* son conexos IGP si existe un camino de *u* a *v* y de *v* a *u* en el grafo IGP.

3.1.1.2 El grafo iBGP

Se representa cada router de un AS por un nodo. Definimos dos conjuntos de routers BGP.

- *S* es el conjunto de los *router de borde* del AS. Estos routers corresponden a las fuentes potenciales de mensajes iBGP en el AS. Serán los que llamaremos nexthops.

- *T* es el conjunto de todos los *routers del AS*. Estos routers corresponden a los objetivos de los mensajes iBGP. Por tanto, concluimos que $S \subseteq T$.

Si se establece una sesión iBGP entre dos routers *u* y *v* se producirán dos enlaces BGP representados como arcos orientados *(u , v)* y *(v , u).* Cada arco *(u , v)* tendrá una etiqueta de acuerdo con el tipo de sesión instalada.

- Si u es un RR de v, se instala la etiqueta DOWN.
- Si u es un cliente del RR v, se instala la etiqueta UP.
- Si no se da ninguna de las situaciones anteriores, se instala la etiqueta OVER. Con ella ambos routers son jerárquicamente iguales.

Llamamos *Libgp = {UP, OVER, DOWN}* a el conjunto de etiquetas iBGP. Como hemos comentado los enlaces BGP siempre aparecen en pares de la siguiente forma:

- UP → DOWN;
- OVER → OVER;
- DOWN → UP.

Podemos observar estas etiquetas en la figura 13, en la que sólo se muestra una etiqueta del par, dado que la otra se puede deducir.

Un grafo iBGP es válido si para todo $s \in S$ y para todo $t \in T$ si existe al menos un camino válido de *s* a *t*, es decir si a todos los routers de borde se puede llegar desde cualquier router del AS.

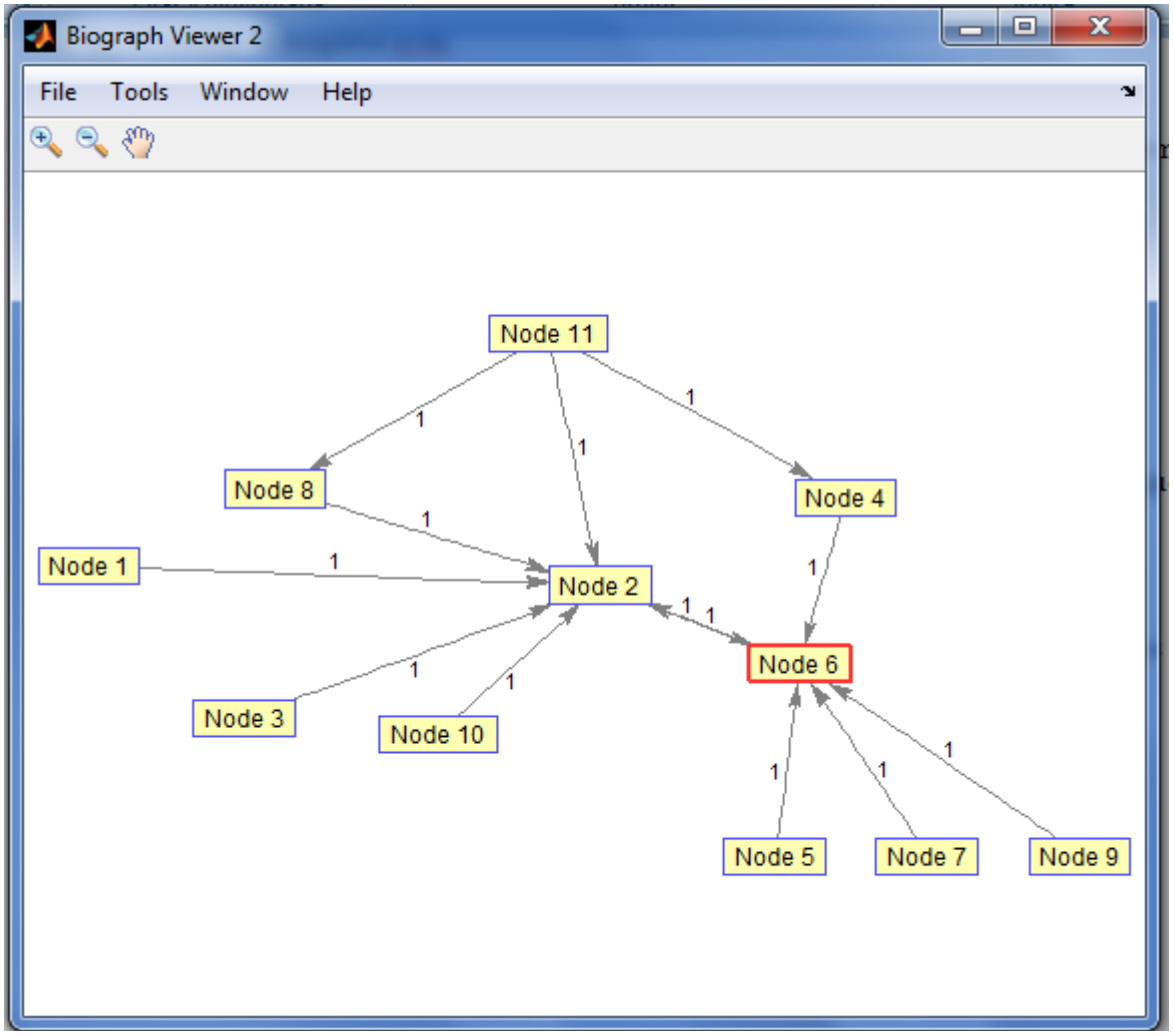


Figura 21: Grafo BGP

En la ilustración superior (figura 21) se muestra un grafo BGP, en el que se pueden observar las sesiones iBGP instaladas. Cada enlace dibujado como una flecha simple, indica una sesión UP. Los enlaces dibujados como una flecha doble indican sesiones de tipo OVER.

3.1.1.3 El grafo extendido

Este grafo se usa para representar cómo se propagan las rutas en BGP. Recordamos aquí los principios básicos de propagación:

1. Si la ruta se recibe de un vecino no cliente, se propaga sólo hacia los clientes.
2. Si la ruta se recibe de un cliente, se propaga a todos los vecinos (clientes o no) excepto al router del cual se ha recibido.
3. Si la ruta se recibe por eBGP, se propaga a todos los vecinos (clientes y no clientes).

Como siempre en BGP, un RR propaga, para cada destino, sólo su mejor ruta.

El *grafo extendido* se construye a partir del grafo BGP, y se utiliza para comprobar que las rutas se difunden de forma correcta. Se explica con todo detalle en Buob et al **[16]**. Aquí trataremos de explicar su construcción y utilización. En la figura 22 se muestra un ejemplo de grafo extendido para una red de 10 nodos (el 11 representa el destino)

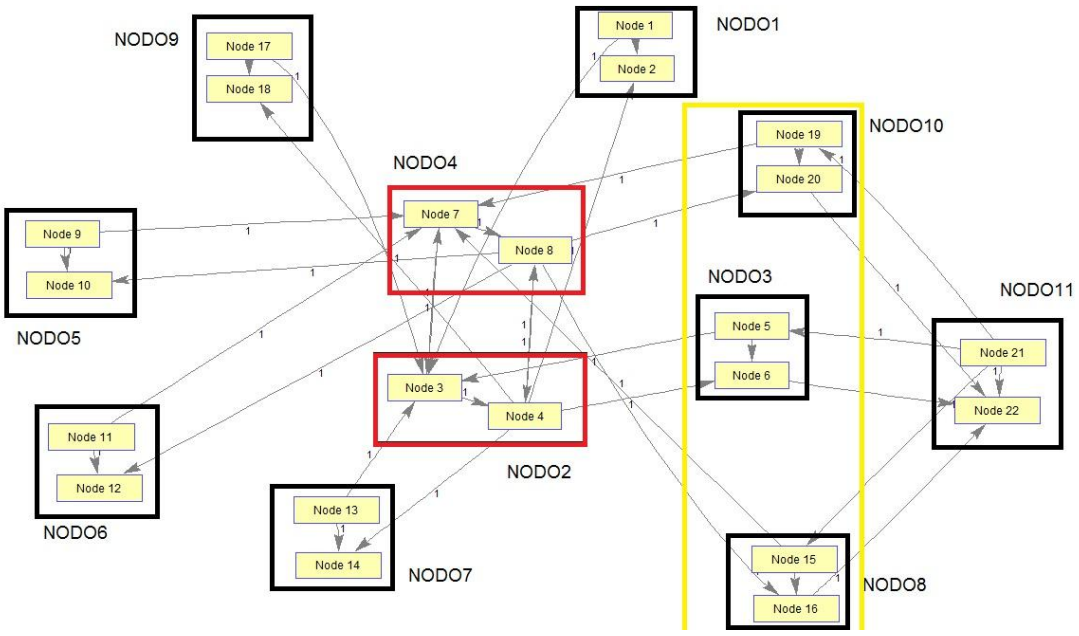


Figura 22: El grafo extendido

Para construirlo, cada nodo se transforma en un metanodo que contiene dos sub-nodos (superior e inferor), los cuales están siempre conectados de arriba abajo (flecha negra en las figuras 23, 24, 25 y 26). Los metanodos se conectan en el grafo extendido según sea su conexión en el grafo BGP, pudiendo ser ésta de dos tipos, mediante una sesión OVER o a través de una sesión UP.

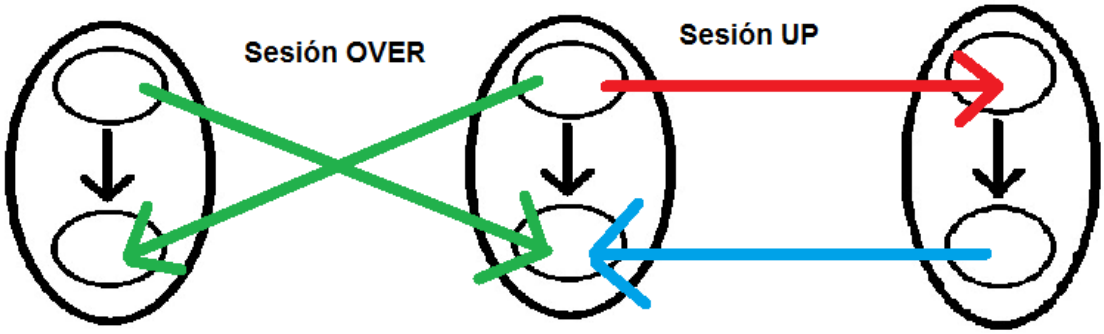


Figura 23: Traducción de sesiones BGP al grafo extendido

En la figura 23 podemos apreciar en qué se traduce cada tipo de sesión BGP en el grafo extendido. Si la sesión es de tipo OVER se conecta el sub-nodo superior de cada metanodo al sub-nodo inferior del otro. Si la sesión es de tipo UP/DOWN se conecta el sub-nodo superior del cliente con el sub-nodo superior del RR y el sub-nodo inferior del RR con el sub-nodo inferior del cliente.

La principal utilidad de este grafo extendido es que gracias a él podemos conocer si una ruta BGP se propaga correctamente hacia cada nodo del AS. Para hacerlo simplemente debemos comprobar la existencia de caminos en el grafo extendido desde el metanodo i al metanodo j (en realidad desde el sub-nodo superior del metanodo i, al subnodo inferior del metanodo j). Si existe este camino, podemos decir que la ruta llega correctamente desde i a j.

Veamos una serie de ejemplos en los que se muestra cómo en el grafo extendido se tienen en cuenta los principios básicos de propagación de rutas cuando se utiliza la reflexión de rutas.

Primera norma: *Si la ruta se recibe de un vecino no cliente, se propaga sólo hacia los clientes.*

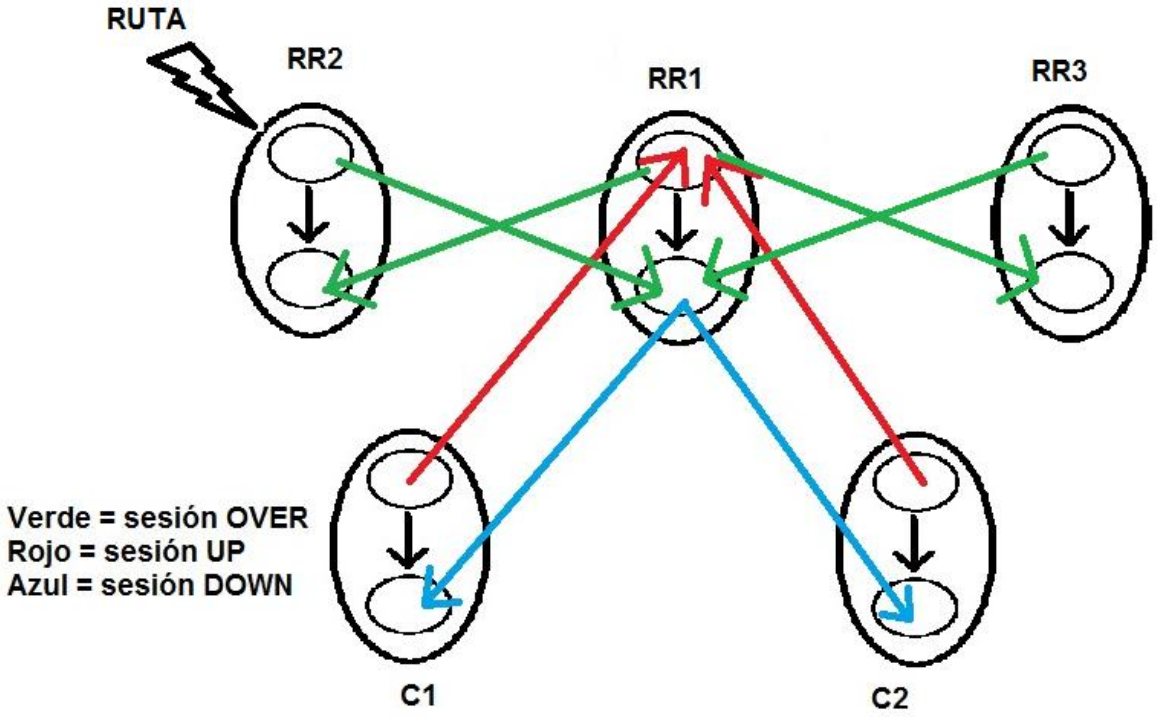


Figura 24: Grafo extendido norma 1

En la figura 24 podemos observar la propagación de una ruta que surge en RR2. Siguiendo la flecha verde llega hasta RR1 (hasta el sub-nodo inferior). Este como ha recibido la ruta por iBGP a través de un nodo no cliente, la debe propagar sólo hacia los clientes, y como vemos en el grafo la única forma de salir del sub-nodo inferior es por las flechas azules, que conducen a los clientes de RR1: C1 y C2. Como era de esperar la ruta no llega a RR3 porque no es cliente de RR1.

Segunda norma: *Si la ruta se recibe de un cliente, se propaga a todos los vecinos (clientes o no).*

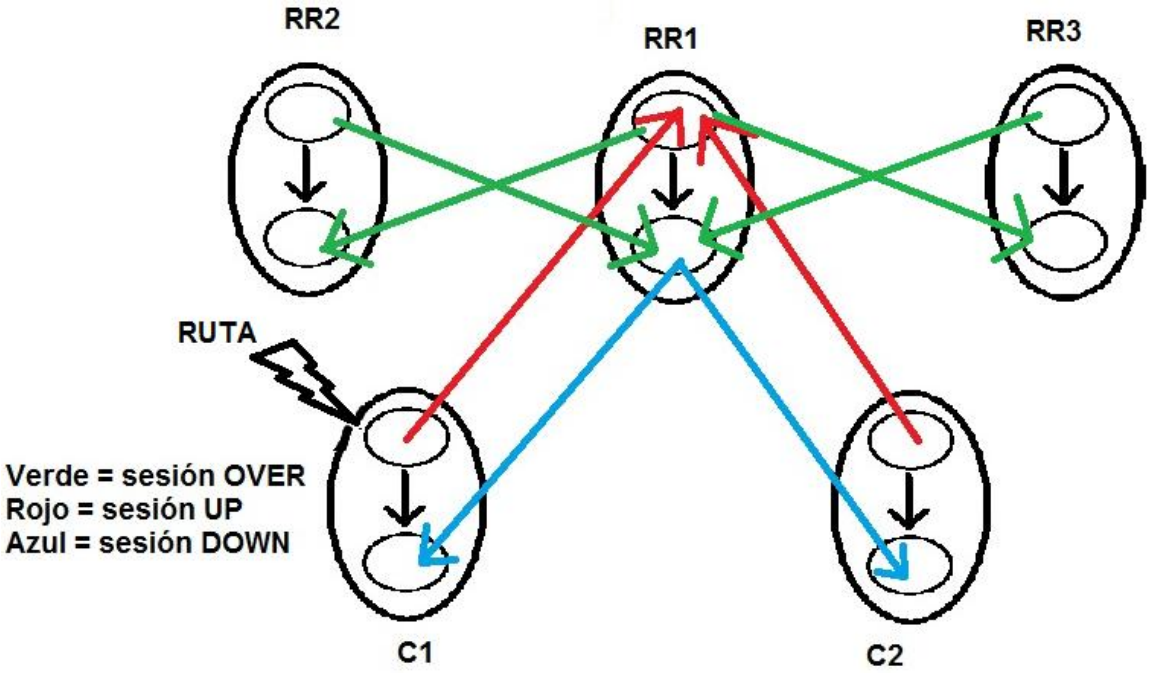


Figura 25: Grafo extendido norma 2

En la figura 25 podemos observar la propagación de una ruta que surge en C1. Siguiendo la flecha roja llega hasta RR1 (hasta el sub-nodo superior), este como ha recibido la ruta por iBGP a través de un nodo cliente, la debe propagar hacia todos los vecinos (clientes o no), y como vemos en el grafo la propaga hacia los routers no clientes, RR2, y RR3, a través de las flechas verdes, y hacia los routers clientes pasando por su sub-nodo inferior (utilizando flecha negra interior a RR1) y utilizando las flechas azules.

Tercera norma: *Si la ruta se recibe por eBGP, se propaga a todos los vecinos (clientes y no clientes).*

Sin necesidad de añadir otra ilustración podemos ver en las figuras 24 y 25 que si una ruta surge en cualquier nodo (es decir le llega por eBGP), es capaz de propagarla hacia todos sus vecinos, ya sean clientes o no.

3.1.2 Matrices utilizadas

Principalmente hemos utilizado cuatro tipos de matrices durante la elaboración de este proyecto:

- La matriz IGP (M_{IGP}), que representa la topología IGP de la red. Cada posición de la matriz muestra el peso del enlace IGP directo entre dos nodos. Así por ejemplo en la posición (i,j) podemos tener:
  - *0*: que indica que no existe enlace IGP entre los dos nodos
  - *Un número positivo*: que indica el peso del enlace IGP que va desde el nodo i al nodo j.

  Un ejemplo de este tipo de matriz (que representa la red de la figura 30, en Resultados) sería:

$$M_{IGP} = \begin{pmatrix} 0 & 27 & 27 & 30 & 30 \\ 27 & 0 & 0 & 30 & 0 \\ 27 & 0 & 0 & 46 & 51 \\ 30 & 30 & 46 & 0 & 47 \\ 30 & 0 & 51 & 47 & 0 \end{pmatrix}$$

- La matriz BGP (M_{BGP}), que representa la topología iBGP de la red. Cada posición de la matriz muestra si existe sesión iBGP entre dos nodos. Así por ejemplo, en la posición (i,j) podemos tener:
  - *0*: que indica que no existe sesión iBGP entre los dos nodos
  - *1*: que indica si existe sesión iBGP y que va desde el nodo i al nodo j.

  Esta matriz generalmente muestra las sesiones de tipo UP (en los resultados mUps o bgpUPfin), un ejemplo de este tipo de matriz se ve en la figura 26. Las sesiones de tipo DOWN (mDowns) se sobre entienden, por lo tanto no es necesario mostrarlas puesto que la matriz de sesiones DOWN es igual a la matriz de sesiones UP traspuesta. Las sesiones OVER no se han tenido en cuenta en la realización del proyecto por simplicidad (ver apartado 3.2.2).

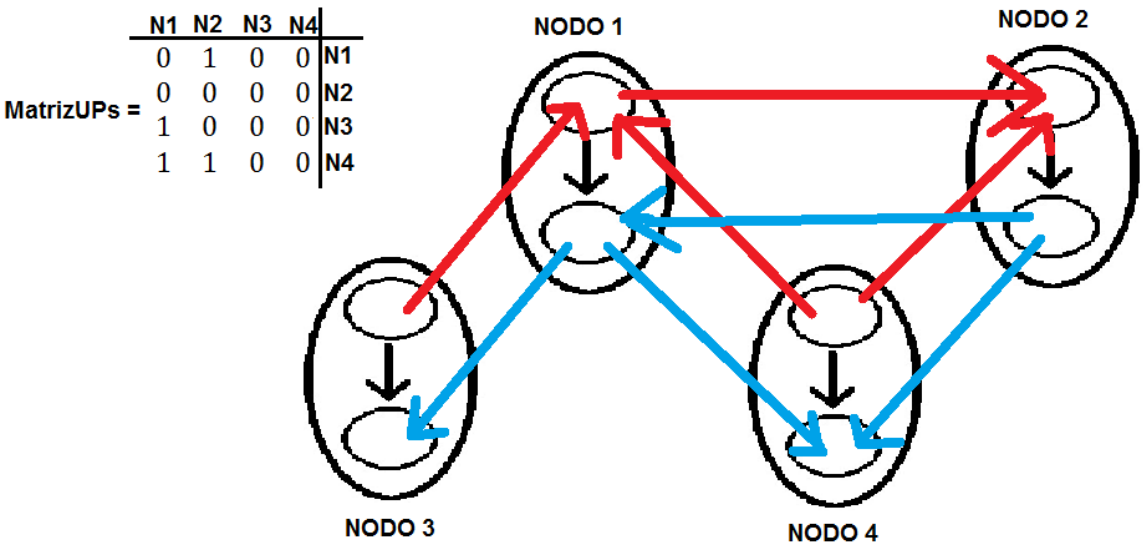


Figura 26: Traducción matriz de Ups (mUps) a grafo extendido

En la figura 26, mostramos un ejemplo de cómo traducir una matriz de UPs (hablamos sobre ella en el siguiente apartado) en el grafo extendido:

- La matriz de distancias IGP (D_{IGP}), que representa la distancia IGP del camino más corto desde cada nodo a todos los demás. Cada posición de la matriz muestra la longitud del camino más corto entre dos nodos. Todas las posiciones de la matriz salvo la diagonal principal (distancia de un nodo a sí mismo) han de ser números positivos menores que infinito (esto indicaría que no existe el camino IGP entre los dos nodos). Podemos encontrar ejemplos de este tipo de matrices en cada una de las pruebas del capítulo Resultados.

3.2 Abordando los objetivos

En este apartado vamos a presentar como hemos alcanzado nuestros objetivos iniciales. Para lograrlos hemos desarrollado una serie de programas utilizando Matlab como herramienta.

3.2.1 ¿Es esta topología fm-óptima?

Para verificar que en una topología se están propagando las rutas de la misma manera que lo haría un full mesh, debemos comprobar si para cada par de nodos router-nexthop (r,n), la ruta se propaga correctamente, de esta forma podemos considerar el par fm-óptimo. Si todos los pares router-nexthop son fm-óptimos entonces podemos considerar la red completa como fm-óptima **[6].**

El algoritmo utilizado para comprobar si es un par es fm-óptimo basado en **[6]** es el siguiente:

1. En primer lugar tenemos que formar el conjunto *Q*, que lo componen los nodos (*n'*) tales que la distancia de *r* a *n*' es mayor que la distancia del *r* a *n:*

$$N \equiv Conjunto\ de\ todos\ los\ los\ nexthop$$

$$R \equiv Conjunto\ de\ todos\ los\ los\ routers$$

$$Q(n,r) = \{n' \in N, dist(r,n') > dist(r,n)\}$$

La función de este conjunto es numerar los posibles next-hops que nos penalizarían si los escogiéramos para el prefijo que estamos estudiando (ya que el mejor nexthop para este prefijo es *n*), es decir, los que nos provocarían un transporte subóptimo. Para estar más seguros, en nuestra programación consideramos que todos los routers del AS son posibles nexthop (gateways) para un determinado prefijo., es decir, *R = N.*

2. Ahora del conjunto de todos los routers, eliminamos todos los que elegirían a algún next-hop de *Q* antes que a *n*, estos serán los routers que podrían ocultar la ruta. Con los router resultantes formamos el conjunto *W.*

$$W(n,r) = \{w \in R | \forall\ n' \in Q(n,r), dist(w,n) < dist(w,n')\}$$

3. Cuando tengamos el conjunto *W* construido, formamos el grafo extendido a partir de él.

4. Finalmente verificamos si le llega la ruta a *r*, es decir si existe un camino de *n* a *r* en el grafo extendido.

Para comprobar la fm-optimalidad en cada par de nodos hemos desarrollado una función llamada comprobarRuta en la que utilizamos el algoritmo que hemos expuesto anteriormente. Si vemos la función mencionada como una caja negra, recibe como *argumentos*:

- *igpMatrix*, que es la matriz que indica la distancia IGP entre cada par de nodos utilizando el camino más corto, es la que hemos llamado D_{IGP}.
- *x*, es simplemente un vector de ceros y unos en el que podemos ver la topología BGP. Se divide en dos partes de igual tamaño, la primera recoge las sesiones de tipo UP (mUps) y la segunda las sesiones de tipo DOWN (mDowns), en caso de que exista sesión se pondrá un 1 en la posición correspondiente en *x*.
- *n*, es el nexthop, el gateway.
- *r*, es el router al que se comprueba se llega la ruta.

La función devuelve 1 o 0 dependiendo de si al router "*r*" le llega la información de rutas BGP de forma correcta desde el nexthop "*n*" (caso en que devuelve 1) o no (caso de respuesta 0).

Como hemos mencionado esta función es de gran utilidad, con ella si conocemos una red IGP con su correspondiente topología BGP podremos saber si ésta es fm-óptima únicamente aplicándola sobre cada par de nodos de la red (en caso de que todas las respuestas sean afirmativas podremos considerar la topología fm-óptima).

Esta función será utilizada en programas posteriores para comprobar la corrección de nuestros resultados. Se han desarrollado dos funciones capaces de aplicar este algoritmo, de esta manera podemos comprobar de dos formas diferentes nuestras soluciones lo que nos lleva a estar más seguros de que éstas son correctas. El código de la más utilizada (la más rápida) puede encontrarse en el anexo A.

3.2.2 Diseño de topologías fm-óptimas

Para diseñar topologías fm-óptimas decidimos comenzar basándonos en el artículo Buob et al. 2008 **[1]**. De esta forma siguiendo sus planteamientos decidimos que las sesiones BGP sólo podrían ser de tipo UP o DOWN, eliminaríamos así las sesiones OVER de nuestras topologías iBGP. Esto se realiza por simplicidad ya que las sesiones OVER hacen que muchas soluciones sean equivalentes lo cual dificulta a la hora de elegir la que consideramos óptima, produciendo una degeneración del problema **[1]**. A modo de ejemplo, en el apartado correspondiente al grafo extendido (3.1.1.3) podemos ver en la figura 23 que utilizando sesiones UP podemos prescindir de las OVER, ya que la propagación en OVER es más restrictiva que en UP: una sesión UP incluye la propagación de rutas de una OVER.

Para la construcción de una topología fm-óptima que minimice el coste IGP asociado a las sesiones iBGP que se instalen necesitamos resolver un problema de optimización binaria (cada sesión BGP existe o no) de gran magnitud. Para solucionarlo hemos replicado el algoritmo utilizado en Buob et al. 2008 **[1]** en el que para resolver el problema utilizan una técnica llamada descomposición de Benders (Benders' decomposition) que permite solucionar problemas de programación lineal muy grandes.

Esta técnica consiste en diferenciar un problema raíz de unos problemas adicionales llamados satélites (figura 29)

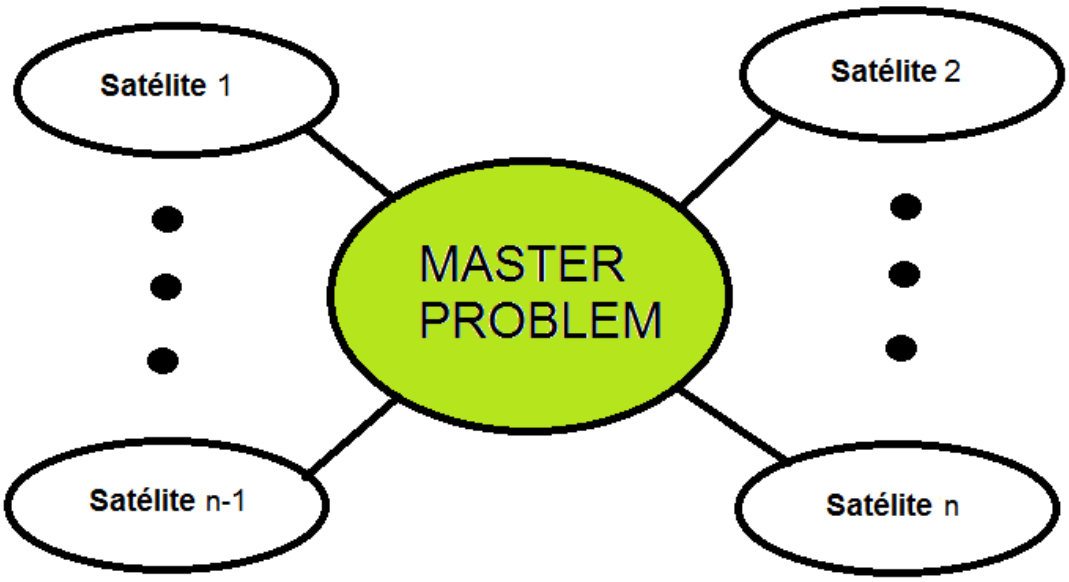


Figura 27: Benders Decomposition

Tras dividir el problema nos disponemos a solucionar únicamente el problema maestro o raíz (master problem). Tras encontrar su solución probamos ésta en cada satélite. En caso de que esta solución no resuelva el problema satélite, se añadirá una restricción adicional a nuestro problema raíz. Pasamos ahora a describir nuestro problema maestro (raíz).

Queremos minimizar el coste IGP asociado a las sesiones iBGP establecidas. Con lo que podemos formular la siguiente función *f*, función que deseamos minimizar:

$$f = \min \left(\sum_{(u,v) \in R} dist(u,v) * (up(u,v) + down(u,v)) \right)$$

En nuestro problema de programación lineal partimos con este set de restricciones:

- Restricciones de desigualdad:

$$\forall\, u, v \in R, up(u,v) + down(u,v) \leq 1$$

- Restricciones de igualdad:

$$\forall\, u, v \in R, up(u,v) = down(v,u)$$

Las primeras restricciones nos indican que entre dos nodos *u* y *v* podemos encontrarnos tres situaciones posibles:

- up *(u, v)* = 0 y down *(u, v)* = 0 → No existe sesión iBGP entre ellos.
- up *(u, v)* = 1 y down *(u, v)* = 0 → Si existe sesión de tipo UP entre *u* y *v*, no puede existir sesión de tipo DOWN entre *u* y *v*.
- up *(u, v)* = 0 y down *(u, v)* = 1 → Si existe sesión de tipo DOWN entre *u* y *v*, no puede existir sesión de tipo UP entre *u* y *v*.

Podríamos pensar en otra situación más, (up *(u, v)* = 1 y down *(u, v)* = 1), pero su aparición está restringida porque en ese caso: up *(u, v)* + down *(u, v)* = 2 > 1. Con esto conseguimos que entre dos nodos u y v, no haya simultáneamente una sesión UP y DOWN lo cual equivaldría a una OVER.

En cuanto a las restricciones de igualdad, nos indican que si existe una sesión de tipo UP entre los nodos *u* y *v*, existe necesariamente una sesión de tipo DOWN entre los nodos *v* y *u*. Esta condición también puede verse de forma recíproca, es decir, si existe una sesión de tipo DOWN entre *u* y *v*, existe necesariamente una sesión de tipo UP entre *v* y *u*. La última observación que podemos extraer de esta restricción es que si no existe sesión UP entre *u* y *v*, tampoco existirá sesión DOWN entre *v* y *u*, y viceversa.

Para realizar esta optimización se ha implementado en Matlab un programa que construye la solución de acuerdo a la técnica mencionada. La estructura del programa es la siguiente, (el programa integro se encuentra en el anexo A).

- Primero definimos la entrada, la red IGP, usando la matriz D_{IGP}.

- Con ella formulamos la función a minimizar objetivo *f, en base a las variables a las variables* M_{UP} y M_{DOWN}.

- El siguiente paso es formular las restricciones iniciales tanto de igualdad como de desigualdad y con ellas ya tenemos el problema maestro (raíz).

- Lo solucionamos mediante la función bintprog de Matlab, que retorna vector que muestra la solución de la optimización.

La solución inicial es la trivial: $M_{UP} = 0$ (todos los elementos de M_{UP} son ceros) que indica que ninguna sesión está instalada.

- Con nuestro problema raíz solucionado pasamos a probar si la solución obtenida también resuelve los satélites, en nuestro caso cada satélite será un par de nodos (i,j).

Para saber si lo resuelve utilizamos la función comprobarRuta entre ese par de nodos, en caso de que la respuesta sea afirmativa la solución será válida, en caso contrario tendremos que añadir una nueva restricción. Esta restricción se forma buscando todos los

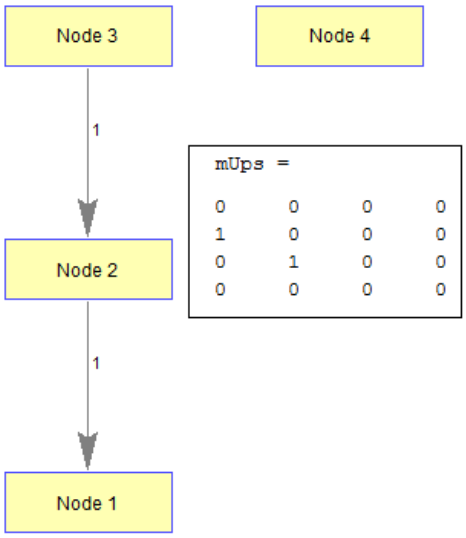
posibles caminos (sesiones) que podrían conectar el par de nodos que estudiamos. Esto lo realizamos añadiendo una sesión provisional a nuestra topología y comprobando si la ruta existe, en caso afirmativo guardamos esta sesión. Realizamos esto con todas las sesiones posibles y vamos guardando todas las soluciones que satisfacen a nuestro par de nodos en estudio, a estas soluciones guardadas las llamamos "sesionesposibles(i,j)" .

- Cuando tengamos todas formamos una restricción única que añadimos a nuestro set de restricciones y solucionamos el problema de nuevo para encontrar la forma óptima de conectar los dos nodos. Cada restricción tiene la forma siguiente:

$$\sum(sesionesposibles(i,j)) \geq 1$$

De esta forma al menos una de las sesiones que conecta al par (i,j) estará en nuestra topología resultante. Para que esto quede más claro a continuación vamos a mostrar un ejemplo sencillo de cómo se formaría la restricción. La red utilizada tiene 4 nodos.

Supongamos que hasta ahora hemos solucionado los pares (1,2) y (1,3) y la topología parcial que tenemos es la que mostramos en la figura28:



Podemos ver que existe camino entre los nodos (1,2) y (1,3). Para asegurar la propagación entre los nodos (1,4), debemos añadir una nueva restricción al problema. Para ello debemos descubrir todos los caminos posibles que satisfacen que exista ruta para el par (1,4). Los caminos a probar son: [(1,4) (2,4) (3,4) (4,1) (4,2) (4,3)].

Figura 28: Ejemplo creando restricción (1)

De este conjunto de posibles sesiones no todas sirven si atendemos a la propagación de rutas, utilizando la función comprobarRuta podemos observar cuales hacen que llegue la ruta de forma correcta y cuáles no. En la figura 29 mostramos por qué algunas sesiones de estas se desechan y otra se aceptan.

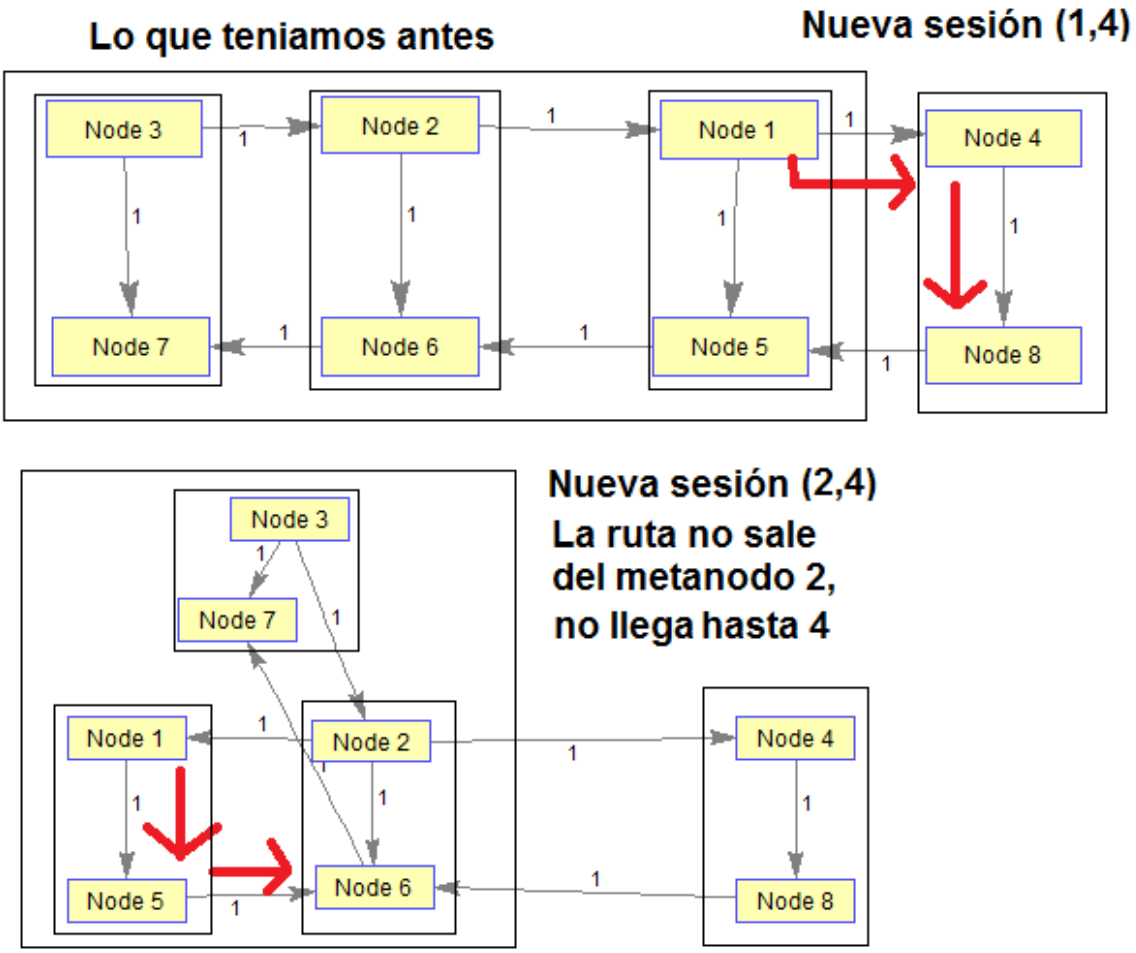


Figura 29: Ejemplo creando restricción (2)

Como vemos con la sesión (1,4) conseguimos que llegue la ruta desde 1 hasta 4, sin embargo con la sesión (2,4) la ruta no llega a su destino, repitiendo esta operación con cada una de las sesiones posibles obtenemos que las únicas que consiguen que llegue la ruta desde 1 hasta 4 son [(1,4) (4,1) (4,2) (4,3)]. Estas 4 sesiones serán nuestras "sesionesposibles". Con ellas formamos la siguiente restricción:

$$UP(1,4) + UP(4,1) + UP(4,2) + UP(4,3) \geq 1$$

Conservaremos esta restricción durante toda la ejecución del problema, asegurándonos de que al menos una de estas sesiones esté en la topología final resultante.

- Realizamos todo este proceso para cada par de nodos.

Al final volvemos a utilizar la función comprobarRuta con cada par de nodos para asegurarnos de que la solución final es correcta. A partir de este momento llamaremos a este algoritmo "*algoritmo BUOB"*, hemos de dejar claro que es una réplica del algoritmo que aparace en Buob et al 2008 **[1].**

Como hemos mencionado en nuestra programación, utilizamos la función de Matlab bintprog que resuelve un problema de optimización en el que las variables son binarias. Observamos que esta función era algo lenta en su ejecución por lo que decidimos indagar en la forma en la que esta función buscaba la solución al problema de optimización. Descubrimos que lo hacía de una forma "demasiado" exhaustiva y que se podía limitar el tiempo de ejecución del programa, a través de un parámetro.

Tras probar su uso nos quedaba la duda de si las soluciones obtenidas con el tiempo limitado serían las mismas que con tiempo ilimitado, en los resultados puede verse que la solución no tenía que ser necesariamente la misma pero el peso de la función que minimizamos si lo era (en los casos probados) con lo que ambas soluciones resultaban equivalentes. De esta forma en nuestra programación asumimos que podemos considerar como buenas las soluciones limitadas por tiempo.

3.3 Mejora del algoritmo de construcción de topologías

El algoritmo que estamos utilizando (replica extraída de Buob et al. 2008 **[1]**) tiene una serie de limitaciones importantes.

En primer lugar como hemos mencionado la utilización de bintprog es bastante lenta cuando se necesita optimizar un número considerable de variables. Aunque este tiempo se puede limitar, parece interesante no hacerlo para poder obtener las mejores soluciones.

En segundo lugar al escalar el problema a redes más grandes, empezaron a surgir problemas de memoria, ya que las matrices utilizadas en el programa adquirían un tamaño inmanejable. Además de los problemas de tiempo y memoria, la función bintprog admite un número finito de variables y al escalar la red hasta el tamaño de 325 nodos (red tier-1 objetivo) el número de variables necesarias excedió el máximo número de variables permitido que es 65535 (2^16 -1). Nosotros utilizábamos 211250 (211250 = 325 x 325 x 2; 325 es el numero de nodos de la red). Con estas limitaciones el número máximo de nodos que podemos manejar es 180 (180 x 180 x 2=64800).

Debido a estas limitaciones decidimos realizar otra implementación diferente que fuese más rápida y eficaz. Por ello decidimos implementar un nuevo algoritmo de tipo greedy. En los algoritmos de este tipo, cuando tomas una decisión acarreas con ella durante el resto de la ejecución, esto traducido a nuestra problemática implica que cuando decidamos que debe existir una sesión, ésta no podrá ya ser excluida de la solución final. El programa completo se encuentra en el anexo A.

Nuestra nueva implementación parte únicamente de una sub-red de dos nodos, para esta red encontramos la mejor forma de conectar estos dos nodos, esta solución se guarda y no se modificara más.

, tomando como mejor manera la que tenga el mínimo coste IGP asociado a las sesiones iBGP que conecten los nodos.

Para hallar esta "mejor manera" tenemos que volver a solucionar un problema de optimización lineal, en el que seguiremos los mismos pasos descritos para la implementación anterior. Utilizaremos la misma función a minimizar *f* y el mismo set de restricciones iniciales pero definidas de otra forma. Una vez tengamos el problema maestro procedemos a aplicar la descomposición de Benders, en esta ocasión tendremos tantos satélites como pares (*nododeestudio*, nodo) es decir *nododeestudio* – 1 satélites (*nododeestudio* es un número que indica cuantos nodos tendrá la red resultante de unir el nodo con el que estamos trabajando a la red anterior).

Realizamos esta operación de forma sucesiva hasta conectar todos los nodos de nuestra red. Llamaremos a este algoritmo "algoritmo greedy".

Con esta implementación el número de variables se reduce de forma significativa. Con el algoritmo anterior en cada iteración manejábamos 2 x (*numero_de_nodos*^2) variables y con el nuevo programa manejamos únicamente 4 x (*nododeestudio*-1) variables, de esta forma con nuestra red de 325 nodos el máximo número de variables a estudiar serían:

$$4 \text{ x } (325 - 1) = 1296 \ll 211250 = 2 \text{ x } (325\text{^}2).$$

Parece evidente que los resultados obtenidos con un algoritmo de este tipo dependerán de manera directa de la ordenación que tomamos a la hora de anexionar los nodos, por esta razón decidimos probar distintas ordenaciones y así ver si obtenemos distintos resultados. En primer lugar probaríamos un número (parámetro modificable) de diferentes ordenaciones escogidas de forma aleatoria, para escoger la solución que tenga menos sesiones iBGP instaladas.

Para obtener el mejor resultado posible con total seguridad, deberíamos probar todas las combinaciones, lo cual implica probar con una gran cantidad de ordenaciones, por ejemplo en la primera red que probamos ordenaciones (red GEANT) contamos con 23 nodos, con esta cantidad de nodos ya es inviable probar todas las ordenaciones posibles porque habría que probar 23! combinaciones. En nuestro caso se han probado únicamente 50 ordenaciones. Los resultados pueden verse en su sección específica, donde también se

expone como obtener buenas ordenaciones de los nodos en función de una serie de parámetros.

Utilizando este algoritmo mejorado conseguimos construir topologías fm-óptimas en menos tiempo que con el algoritmo expuesto en Buob et al. 2008 [1]. Además de esto, si utilizamos una ordenación correcta los resultados serán mejores que los obtenidos mediante el citado algoritmo.

3.4 Añadiendo redundancia

Hasta este momento nuestras implementaciones eran capaces de crear topologías fm-óptimas a partir de la red IGP. Con el programa que añade redundancia otorgamos una propiedad adicional a nuestras soluciones, ahora además de fm-óptimas podemos considerarlas robustas.

Para aplicarles esta robustez, nuestra idea consistía en que teniendo la red sin redundancia ya construida, empezar a provocar fallos en enlaces y comprobar si perdíamos la fm-optimalidad, en caso de que se perdiera tendríamos que añadir alguna sesión más para cumplir con los requisitos.

En lugar de simular fallos en enlaces aislados decidimos simular fallos en nodos, lo cual es factible de ocurrir en una red real y además es una forma de simular fallos en enlaces de forma simultánea (lo que otorga más fiabilidad a la red resultante), ya que si falla un nodo la situación final es la misma que si fallan todos los enlaces que lo involucran.

De esta forma, nuesto algoritmo es muy sencillo:

- Cargamos la topología BGP sin redundacia
- Recorremos todos los nodos de la red haciendo las siguientes acciones en cada uno:
    - Provocamos fallo en el nodo
    - Comprobamos en qué pares de nodos hemos perdido la fm-optimalidad
    - Creamos nuevas sesiones para hacer que estos pares vuelvan a ser fm-óptimos. Estas sesiones serán directas entre los nodos entre los que se está comprobando la ruta.

4

Resultados

4.1 Comprobación de rutas y topologías

Para empezar vamos a mostrar cómo actúa la función de comprobar ruta en una red pequeña (5 nodos) para poder entender su funcionamiento. Esta red ha sido creada de forma aleatoria. La red IGP y la topología BGP se muestran en la figura 30.

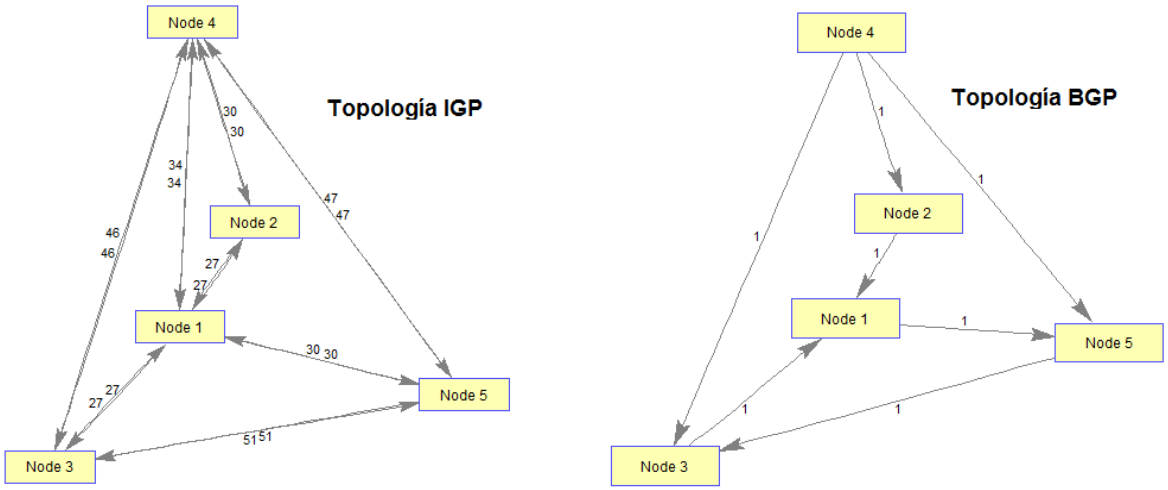


Figura 30: Probando comprobarRuta, red IGP y topología BGP

Por ejemplo vamos a comprobar si al router 4 le llega una ruta que parta del nodo 1 (r=4, n=1).

Utilizamos nuestra función:

```
>> comprobarRuta(igpMatrix,x,1,4)

ix =

     3     5

ans =

     1
```

- "*ix*" es el conjunto de nodos *Q*. Recordemos que este conjunto lo forman los nodos que están más lejos que el nexthop (nodo 1) partiendo desde el router (nodo 4). Asi dist(4,1) = 34, dist(4,2) = 30, dist(4,3) = 46 y dist(4,5)= 47;
- *ans*, es la repuesta del programa.

La función responde "1", con lo que la ruta llega correctamente. Veamos ahora el proceso para llegar a este resultado. El siguiente paso es crear el conjunto *W* (nodos que eligen al nexthop antes que a cualquier nodo de *Q*).

Nodo 1: dist(1,1) = 0, dist(1,3) = 27 y dist(1,5)= 30; Forma parte de W
Nodo 2: dist(2,1) = 27, dist(2,3) = 54 y dist(2,5)= 57; Forma parte de W
Nodo 3: dist(3,1) = 27, dist(3,3) = 0 y dist(3,5)=51; No forma parte de W
Nodo 4: dist(4,1) = 34, dist(4,3) = 46 y dist(4,5)= 47; Forma parte de W
Nodo 5: dist(5,1) = 30, dist(5,3) = 51 y dist(5,5)= 0; No forma parte de W

De esta forma el conjunto *W* es [Nodo 1, Nodo 2, Nodo 4];

Ahora debemos borrar de nuestra solución BGP las conexiones en las que interviene algún nodo que no forme parte de *W*, es decir borramos las sesiones en las que participan el Nodo 3 y el Nodo 5.

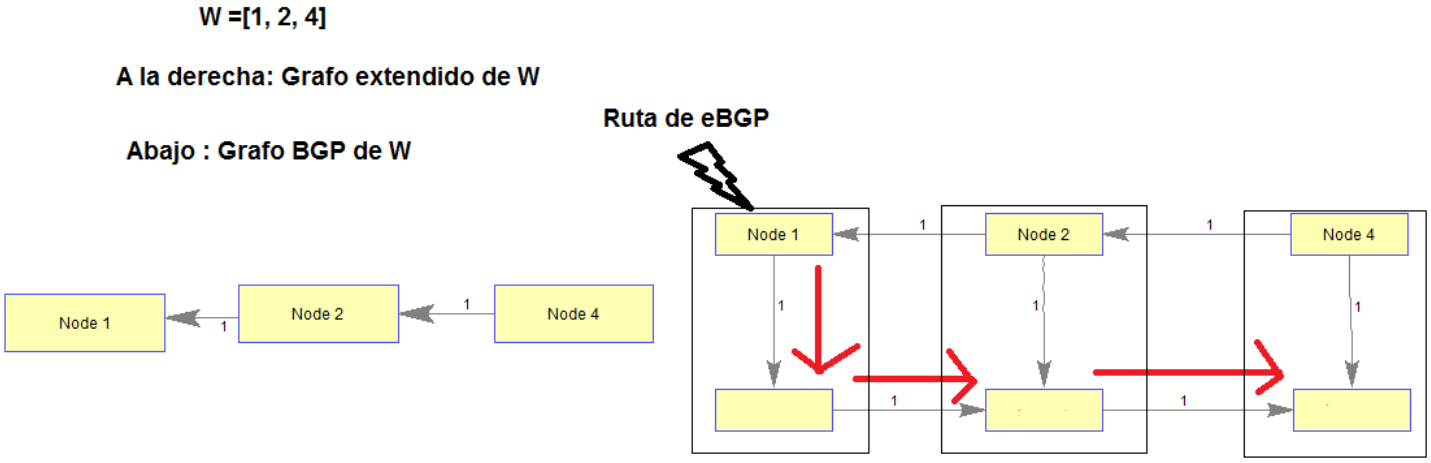


Figura 31: Propagación de ruta 1 → 4

En el grafo extendido (a la derecha en la figura 31) vemos que existe camino desde el sub-nodo superior del metanodo 1, al sub-nodo inferior del metanodo 4. Como existe este camino entonces la ruta llega correctamente desde el nexthop (nodo 1) al router (nodo 4).

Ahora que ya sabemos cómo funciona esta importante función vamos a presentar los resultados obtenidos con los programas desarrollados explicando cómo se han conseguido

en cada caso. Esto nos permite comprobar si una topología es fm-óptima o no, únicamente comprobando todos los pares *(r,n)*.

4.2 Resultados algoritmo BUOB

Recordemos que este programa era capaz de crear una topología BGP para una red IGP dada. Como hemos podido ver en el apartado de diseño podemos instalar restricciones temporales a la ejecución. Recomendamos utilizar el programa sin restricciones temporales únicamente en redes de tamaño pequeño (menos de 10 nodos).

La primera red que utilizaremos será una red de 4 nodos creada por nosotros, mediante una función que hemos desarrollado. Mediante esta función creamos topologías IGP en las que lo único que determinamos es el número de routers (que por otro lado también podemos variar). Todos los enlaces entre routers pueden existir o no y en caso de existir su peso asignado es aleatorio. Las redes creadas con esta función serán simétricas (peso(u,v) = peso(v,u)). Esta función la utilizaremos para crear las redes de 4, 5 y 6 nodos que utilizamos a lo largo de esta sección.

Para cada una de estas redes, construimos la matriz de distancias IGP (D_{IGP}) a partir de su matriz IGP (M_{IGP}). A modo de ejemplo supongamos que la matriz de distancias IGP es la siguiente:

$$D_{IGP} = \begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 2 & 1 \\ 3 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}$$

Como vemos la diagonal principal es 0, la distancia de un nodo a sí mismo es 0, y como habíamos advertido la matriz es simétrica puesto que nuestra red IGP también lo es.

Si aplicamos el algoritmo a esta red obtenemos la siguiente solución:

```
Topologia final:

mUps =

     0     1     0     0
     0     0     0     1
     0     1     0     0
     0     0     1     0


mDowns =

     0     0     0     0
     1     0     1     0
     0     0     0     1
     0     1     0     0

Elapsed time is 0.634529 seconds.
Biograph object with 4 nodes and 4 edges.

iteraciones =

    35
```

Figura 33: Resultados algoritmo BUOB (4 nodos), consola de Matlab

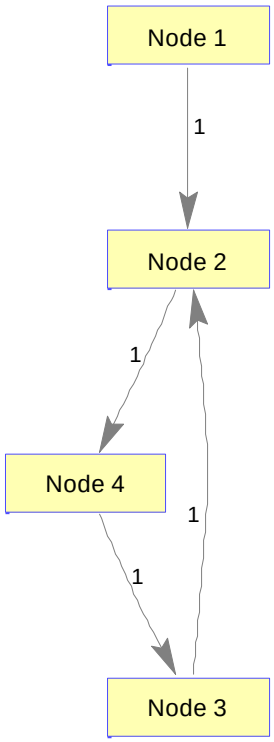


Figura 32: Resultados algoritmo BUOB (4 nodos), topología BGP

Podemos observar que tiempo de ejecución es de 0.6 segundos.

La topología resultante obtenida consta únicamente de cuatro sesiones, esto puede verse de forma clara tanto en el grafo en el que sólo hay cuatro flechas como en la matriz de resultados en las que solo hay cuatro “1”s.

En este caso la reducción con respecto al full mesh en el número de sesiones es de:

$$\frac{4}{4\ x\ 3} = \frac{4}{12} = \frac{1}{3}$$

Como vemos conseguimos una topología fm-optima con un tercio de las sesiones necesarias en un full mesh.

Durante todo el proyecto para realizar el cálculo de la reducción seguimos el método empleado en Buob et al. 2008**[1]**, lo cual nos permite comparar los resultados obtenidos con los suyos. De esta forma contamos las sesiones del full mesh como dobles considerando que en realidad son dos conexiones, de *u* a *v* y de *v* a *u*, son las que hemos estado llamando OVER.

A continuación veamos un ejemplo de una red de 5 nodos, en este caso la matriz de distancias IGP es:

$$D_{IGP} = \begin{pmatrix} 0 & 4 & 5 & 2 & 5 \\ 4 & 0 & 8 & 3 & 6 \\ 5 & 8 & 0 & 7 & 2 \\ 2 & 3 & 7 & 0 & 7 \\ 5 & 6 & 2 & 7 & 0 \end{pmatrix}$$

Resultados:

```
Topologia final:

mUps =

     0     0     0     0     0
     1     0     0     0     0
     1     0     0     0     0
     1     1     0     0     0
     0     1     1     0     0


mDowns =

     0     1     1     1     0
     0     0     0     1     1
     0     0     0     0     1
     0     0     0     0     0
     0     0     0     0     0

Elapsed time is 2.368892 seconds.
Biograph object with 5 nodes and 6 edges.
```

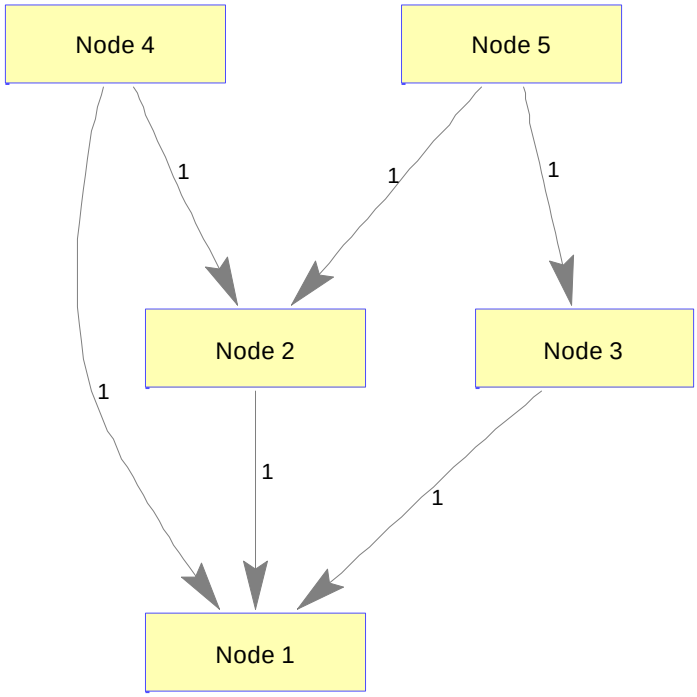


Figura 34: Resultados algoritmo BUOB (5 nodos), consola de Matlab

Figura 35: Resultados algoritmo BUOB (5 nodos), topología BGP

Reducción en el número de sesiones con respecto a full mesh:

$$\frac{46}{5\ x\ 4} = \frac{6}{20} = \frac{3}{10} = 0{,}3$$

En la siguiente tabla mostramos el valor del peso asociado a las sesiones instaladas, así como su suma. La última columna muestra el peso asociado que tendríamos instalando full mesh:

Sesión (origen, destino)	Peso IGP	Peso IGP del full mesh
(2,1)	4	.
(3,1)	5	.
(4,1)	2	.
(4,2)	3	.
(5,2)	6	.
(5,3)	2	.
SUMA	**22**	**98**

Tabla 1: Buob, Suma de pesos IGP (5 nodos)

Finalmente probamos con una red de 6 nodos en la que podemos observar el cambio en la topología si restringimos demasiado el tiempo de ejecución de la función bintprog. La primera prueba es sin restricción temporal y la segunda con un tiempo máximo de 1 segundo para cada ejecución de la función bintprog.

$$D_{IGP} = \begin{pmatrix} 0 & 6 & 6 & 4 & 5 & 5 \\ 6 & 0 & 4 & 4 & 4 & 7 \\ 6 & 4 & 0 & 8 & 8 & 11 \\ 4 & 4 & 8 & 0 & 3 & 3 \\ 5 & 4 & 8 & 3 & 0 & 4 \\ 5 & 7 & 11 & 3 & 4 & 0 \end{pmatrix}$$

Primera prueba (dejamos converger a bintprog).

```
Topologia final:

mUps =

     0     1     1     0     0     1
     0     0     0     0     1     0
     0     1     0     0     0     0
     1     1     0     0     0     0
     0     0     0     1     0     1
     0     0     0     1     0     0


mDowns =

     0     0     0     1     0     0
     1     0     1     1     0     0
     1     0     0     0     0     0
     0     0     0     0     1     1
     0     1     0     0     0     0
     1     0     0     0     1     0

Elapsed time is 157.744457 seconds.
Biograph object with 6 nodes and 10 edges.
```

Figura 36: Resultados algoritmo BUOB (6 nodos SR), consola de Matlab

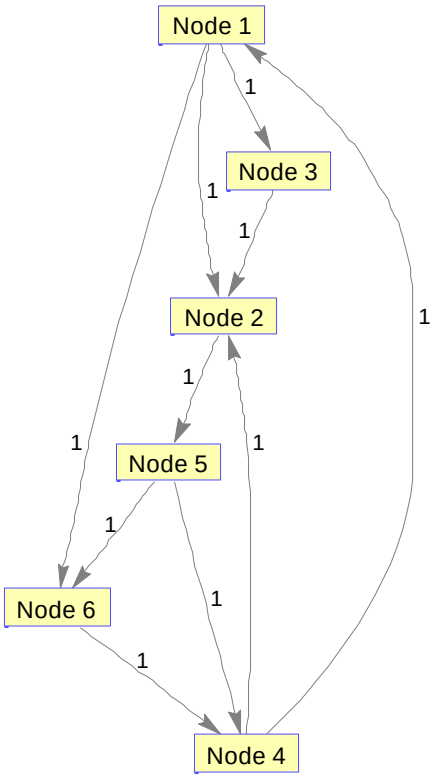


Figura 37: Resultados algoritmo BUOB (6 nodos SR), topología BGP

Reducción con respecto a full mesh: $\frac{10}{6\ x\ 5} = \frac{10}{30} = \frac{1}{3} = 0{,}33$

Al igual que en el apartado anterior mostramos la tabla con los costes IGP asociados a las sesiones instaladas, así como el coste IGP total que tendríamos en full mesh:

Sesión (origen, destino)	Peso IGP	Peso IGP del full mesh
(4,1)	4	.
(1,2)	6	.
(3,2)	4	.
(4,2)	4	.
(1,3)	6	.
(5,4)	3	.
(6,4)	3	.
(2,5)	4	.
(1,6)	5	.
(5,6)	4	.
SUMA	**43**	**164**

Tabla 2: Buob, Suma de pesos IGP (6 nodos) sin restricción temporal

Segunda prueba (restringimos el tiempo de ejecución de bintprog a 1 segundo).

```
Topologia final:

mUps =

     0     0     0     0     0     0
     1     0     0     0     0     0
     1     1     0     0     0     0
     1     1     0     0     0     0
     0     1     0     1     0     0
     1     0     0     1     1     0


mDowns =

     0     1     1     1     0     1
     0     0     1     1     1     0
     0     0     0     0     0     0
     0     0     0     0     1     1
     0     0     0     0     0     1
     0     0     0     0     0     0

Elapsed time is 4.772762 seconds.
Biograph object with 6 nodes and 10 edges.
```

Figura 39: Resultados algoritmo BUOB (6 nodos CR), consola de Matlab

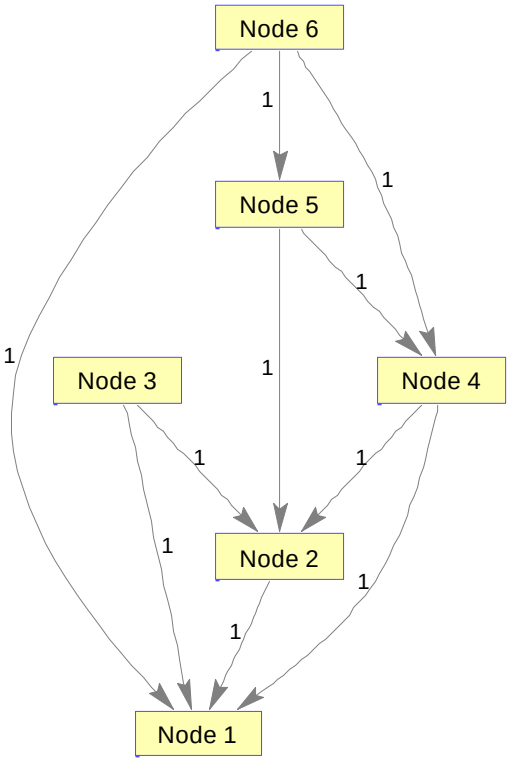


Figura 38: Resultados algoritmo BUOB (6 nodos CR), topología BGP

La reducción con respecto a full mesh en el número de sesiones, es la misma que en la primera prueba, sin embargo, sí hemos conseguido una importante reducción temporal pasando de 157 segundos a menos de 5 segundos.

Sesión (origen, destino)	Peso IGP	Peso IGP del full mesh
(4,1)	4	.
(2,1)	6	.
(3,1)	6	.
(6,1)	5	.
(3,2)	4	.
(4,2)	4	.
(5,2)	4	.
(5,4)	3	.
(6,4)	3	.
(6,5)	4	.
SUMA	**43**	**164**

Tabla 3: Buob, Suma de pesos IGP (6 nodos) con restricción temporal

Como vemos en la tabla 3 el resultado de la suma es el mismo pero hemos obtenido diferente topología, esto si lo extrapolamos a redes más grandes nos indica que al restringir el tiempo es posible que nuestras soluciones no sean las mejores posibles, aunque si serán fm-optimas.

Además en estos resultados podemos observar que al restringir el tiempo se produce un sesgo a favor de las sesiones cuyo nodo destino es un número bajo, en el ejemplo expuesto arriba observamos que el nodo 1 tiene ahora 4 clientes mientras que sin la restricción temporal tenia únicamente 1. Una posible explicación de por qué esto se produce podría basarse en que la función de optimización bintprog trabaja con un resultado en forma de árbol con lo que si le cortamos el tiempo le impedimos comprobar todas las ramas. Por ello, las ramas situadas más a la derecha del árbol (nodos numerados mayores) no se comprueban con la misma frecuencia que los nodos de la rama izquierda (1,2…). Por esto es más probable que los nodos de menor orden reciban más sesiones que los de mayor orden.

4.2.1 Resultados red GEANT

Antes de mostrar los resultados obtenidos, vamos introducir cómo es esta red. La red GEANT es la principal red multi-gigabit europea de ordenadores cuyo uso está destinado para la investigación y la educación. La red opera a gran velocidad, el mínimo es 155 Mbit/s y el máximo se encuentra en torno a los 10 Gbit/s en la red de fibra óptica del núcleo.

En la actualidad se pretende conectar a GEANT (realmente GEANT2) con otras redes regionales de investigación (como JANET, HEAnet, red de Abilene, CANARIE, ESnet, SINET) para crear una sola red global de investigación.

Para mostrar el tamaño de esta red vamos a fijarnos en su evolución, la figura 40 es una imagen extraída de la página principal de GEANT en la que se muestra su extensión en el año 2001:

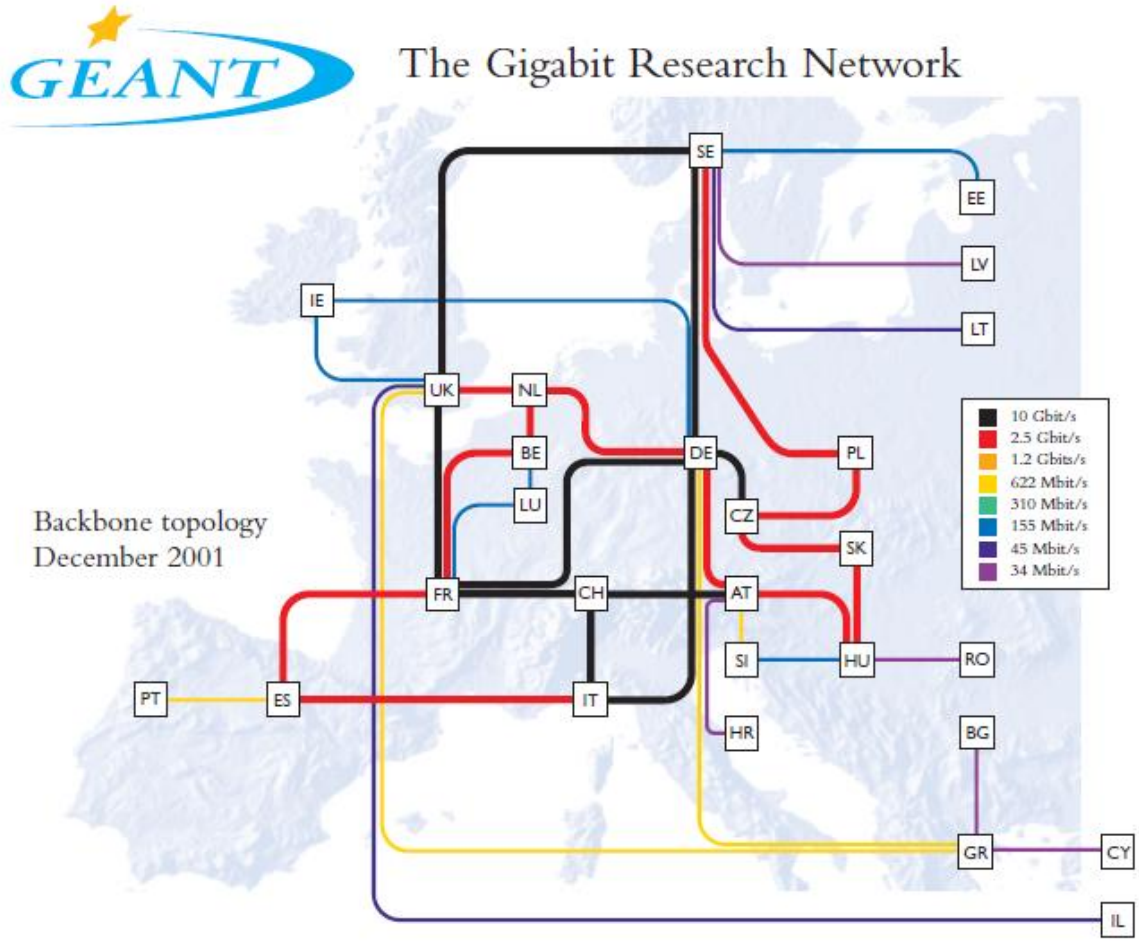


Figura 40: Red GEANT 2001

Como vemos en sus inicios constaba de muy pocas líneas de 10 Gbit/s, en la siguiente imagen, (figura 41) podemos ver como el número de este tipo de líneas se amplió llegando a cubrir gran parte de Europa con ellas.

Con esta red se conectan más de 30 países con sólo 25 PoPs (puntos de presencia, nuestros nodos).

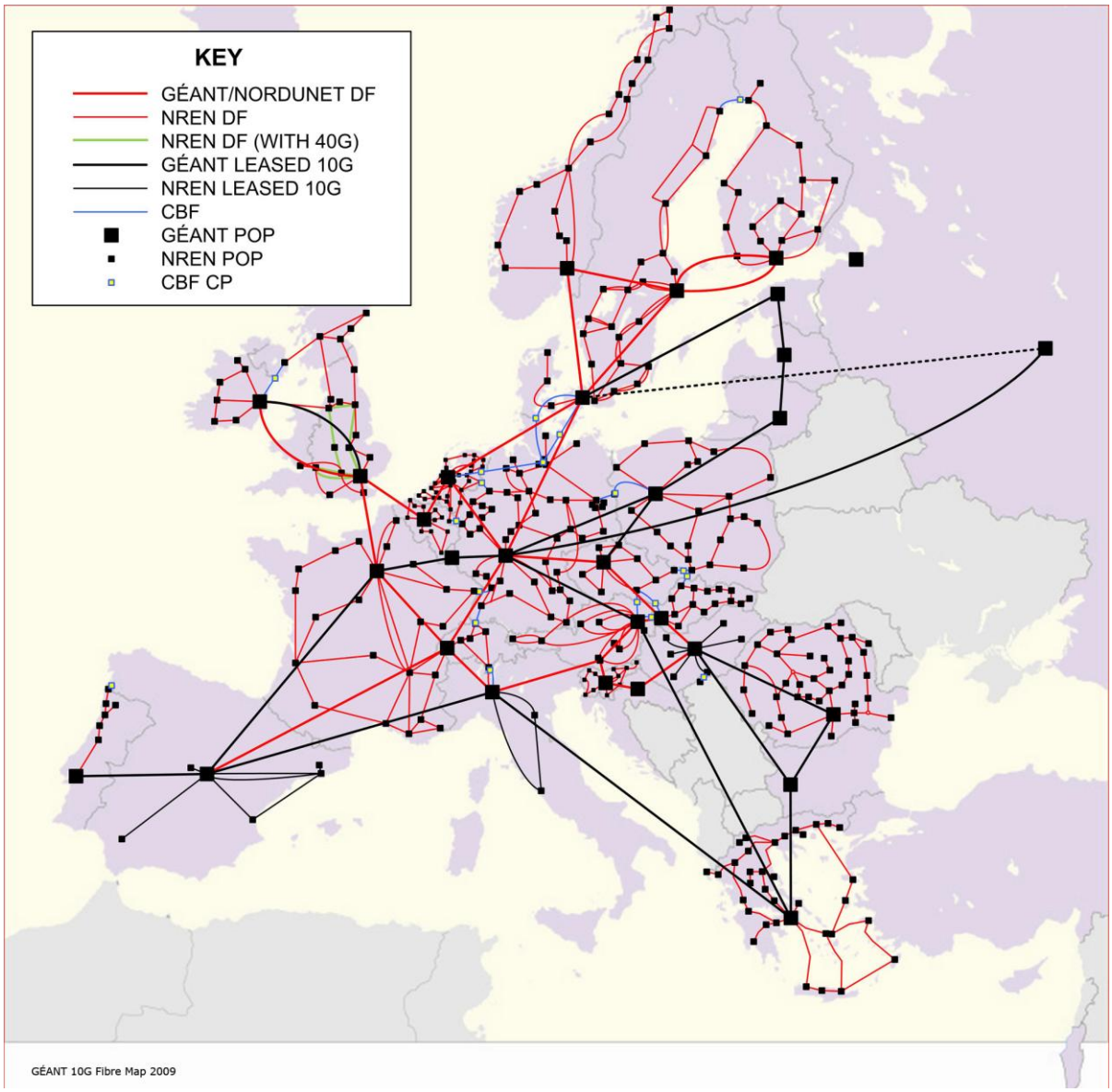


Figura 41: Red GEANT 2009

Una vez introducida, vamos a explicar "nuestra" red GEANT, su topología se ha obtenido de la pagina web de *igen generator* **[12].** De esta web se pueden extraer tanto ésta como otras redes, por ejemplo la red Abilene que hemos mencionado anteriormente y que utilizaremos más adelante. La topología de GEANT que estamos utilizando tiene 23

nodos (no es la topología actual, esta algo desfasada) conectados entre sí a través de 74 enlaces IGP.

Tras ejecutar el programa (utilizando 1 segundo como restricción temporal para la ejecución de bintprog) obtenemos la topología BGP mostrada en la figura 42. Como podemos ver de este grafo es bastante complicado poder extraer conclusiones, por lo que decidimos mostrar las conexiones en la tabla 4.

El tiempo de ejecución ha sido: 77.094347 segundos.

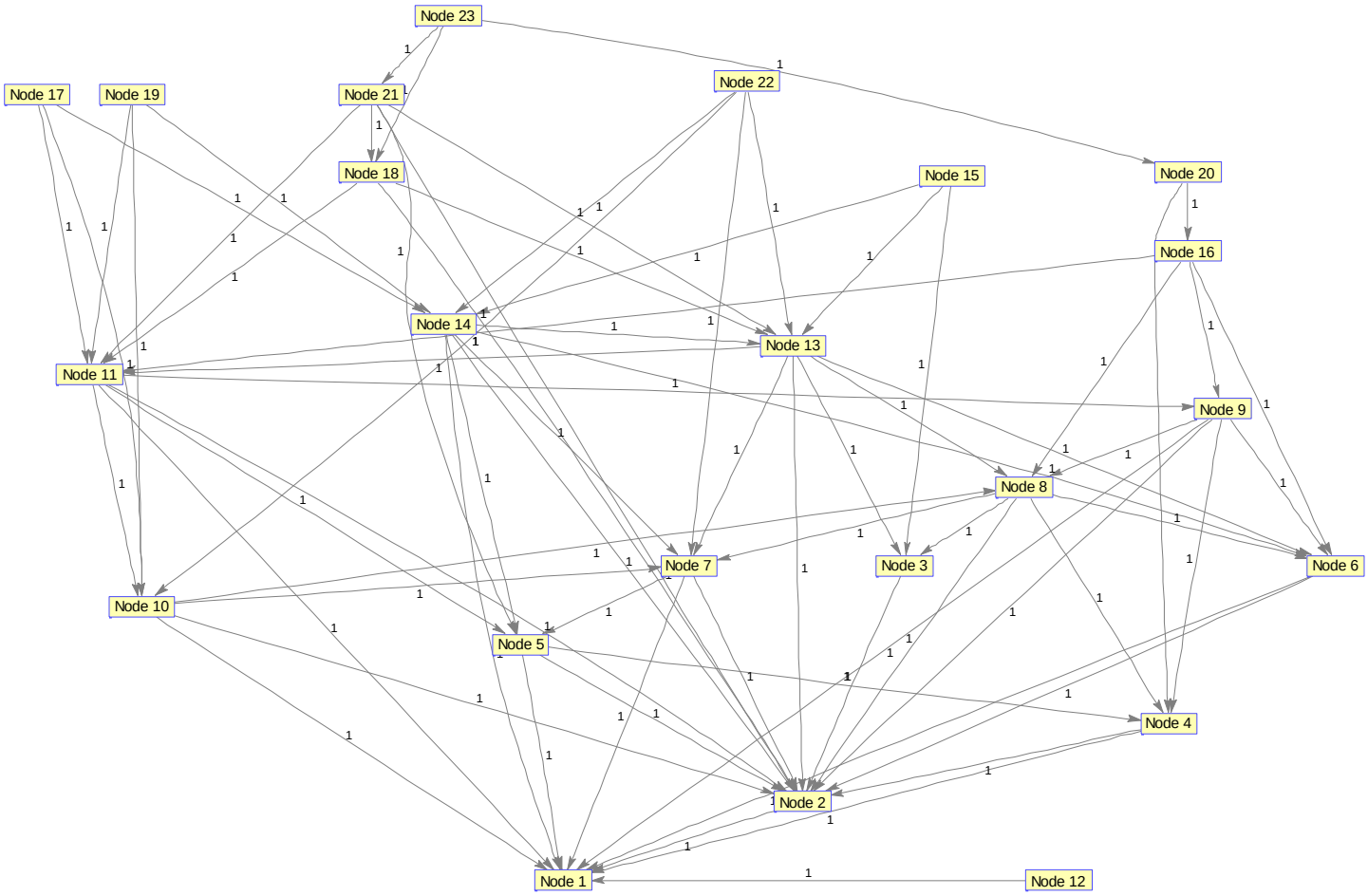


Figura 42: Resultados GEANT, grafo topología final

En total tenemos 74 sesiones iBGP instaladas con las que conseguimos que su funcionalidad sea igual a la que tendría con un mallado total (full-mesh), en cuyo caso el número de sesiones establecidas seria: 23 x 22 = 506 sesiones

Podemos observar que la reducción es muy importante, concretamente para este caso tenemos casi siete veces menos sesiones que las que necesitaríamos utilizando la topología del full-mesh.

$$\frac{74}{506} = 0.15 \qquad \frac{506}{74} = 6.84$$

Origen	Destino	Origen	Destino	Origen	Destino	Origen	Destino
2	1	**8**	3	**9**	8	**15**	14
5	1	**12**	3	**10**	8	**17**	14
7	1	**15**	3	**13**	8	**19**	14
4	1	**5**	4	**16**	8	**22**	14
6	1	**8**	4	**11**	9	**20**	16
9	1	**9**	4	**16**	9	**21**	18
10	1	**20**	4	**11**	10	**23**	18
11	1	**7**	5	**17**	10	**23**	20
12	1	**11**	5	**19**	10	**23**	21
14	1	**14**	5	**22**	10		
3	2	**21**	5	**13**	11		
4	2	**8**	6	**16**	11		
5	2	**9**	6	**17**	11		
6	2	**12**	6	**18**	11		
7	2	**14**	6	**19**	11		
8	2	**16**	6	**21**	11		
9	2	**8**	7	**14**	13		
10	2	**10**	7	**15**	13		
11	2	**13**	7	**18**	13		
13	2	**14**	7	**21**	13		
14	2	**22**	7	**22**	13		
18	2						
21	2						

Tabla 4: Sesiones de red Geant

En esta tabla (tabla 4) podemos volver a observar que los nodos iniciales, participan en más sesiones iBGP que el resto, esto puede deberse a la restricción temporal de la función bintprog, ya que como mencionamos en el apartado anterior, al restringir el tiempo de la ejecución de bintprog se produce un sesgo a favor de las sesiones hacia los nodos de numeración menor.

Como hemos dicho conseguimos una reducción muy importante en el número de sesiones sin perder funcionalidad, lo cual es el objetivo de este proyecto. Otros autores han trabajado también con esta red obteniendo resultados similares, por ejemplo en el artículo del que obtuvimos el algoritmo **([1])** para la red GEANT obtienen funcionalidad completa con 74 sesiones de tipo UP. Es decir, utilizan las mismas sesiones que nosotros, pero su red utilizada es más pequeña, tiene 22 nodos.

Estos resultados han sido obtenidos poniendo como límite temporal a la función bintprog de 1 segundo, ahora vamos a probar a subir este tiempo para ver si observamos variaciones significativas. Las sesiones cambiarán, pero nos interesa saber si también lo hará el coste IGP total asociado a éstas.

El coste total IGP asociado a la topología full mesh en GEANT es de: 1.644.048.

Para 1 seg tenemos un coste total asociado de 198.755 con 74 sesiones.

Para 20 seg tenemos un coste total asociado de 198.755 con 74 sesiones.

No podemos asegurar que en caso de no restringir el tiempo la solución sea la misma, pero sí parece que al menos será semejante.

4.2.2 Resultados Red Abilene

Abilene Network es una red de alto rendimiento que sirvió como backbone (o red troncal) de Internet2. Ahora forma parte de The Internet2 Network. Cuando hablamos de Internet2 nos referimos a un consorcio sin ánimo de lucro que desarrolla tecnologías de redes de alta velocidad. Además de esto han desarrollado tecnologías relacionadas con IPv6, IP multicast y calidad del servicio (QoS).

Más de 220 instituciones participan en Abilene, la mayoría universidades, pero también corporaciones e instituciones afiliadas en todo EE.UU y Puerto Rico.

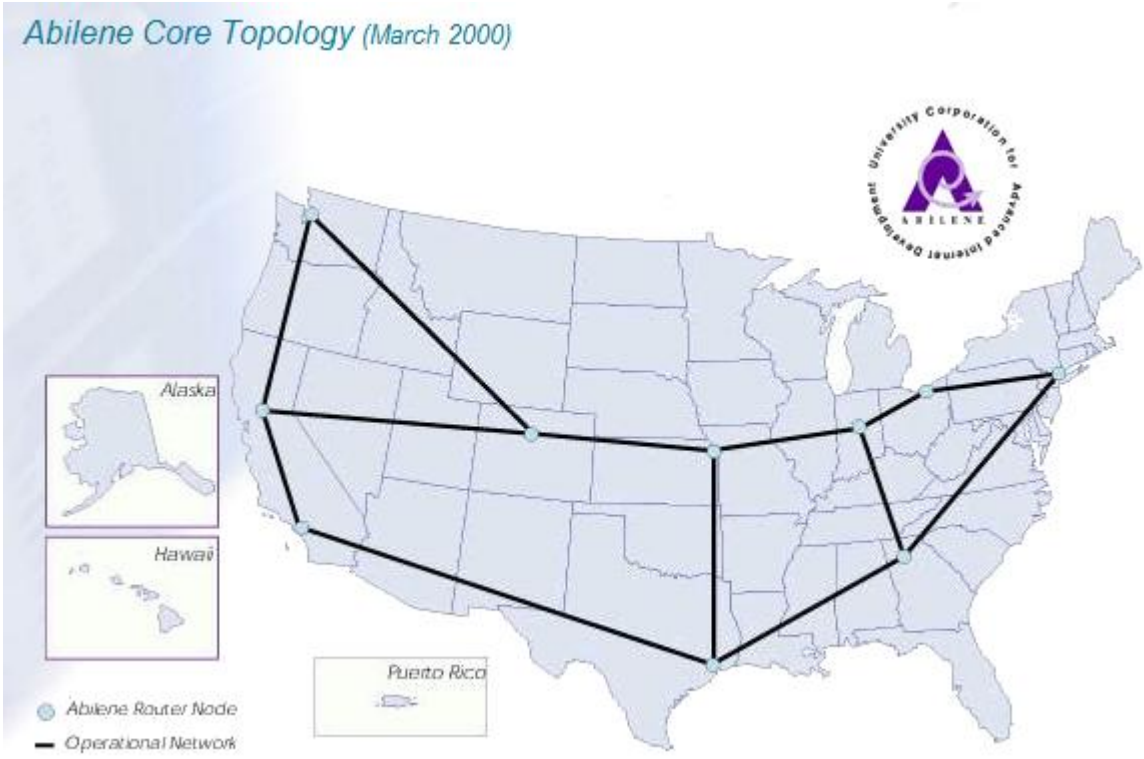


Figura 43: Topología Red Abilene

La red Abilene fue desarrollada por una colaboración entre Qwest Communications, Nortel, Juniper Networks, Cisco Systems y la Universidad de Indiana.

La red Abilene se inició en 1999 con una velocidad de 2,5 gigabits por segundo. En 2003 comenzó una actualización para funcionar a 10 gigabits por segundo, la cual fue completada y anunciada el 4 de febrero de 2004.

Entre 2006 y 2007 se dejó de usar el nombre Abilene en favor de The Internet2 Network, tras una actualización en su infraestructura. El objetivo de la Internet2 Network es llegar a ofrecer 100 gigabits por segundo de velocidad entre nodos.

Esta red cuenta con 11 nodos, su topología IGP la hemos obtenido de la página web de Igen Generator **[12].** En la figura 43 mostramos un esquema de la misma.

Tras comentar brevemente las características de la red vamos a mostrar nuestros resultados.

```
Topologia final:

mUps =

     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     1     0     1     0     0     0     0
     0     0     0     0     1     0     1     1     0     0     0
     0     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     0     0     1     1     1     0

Elapsed time is 16.561645 seconds.
Biograph object with 11 nodes and 19 edges.
```

Figura 44: Resultados Abilene, consola de Matlab

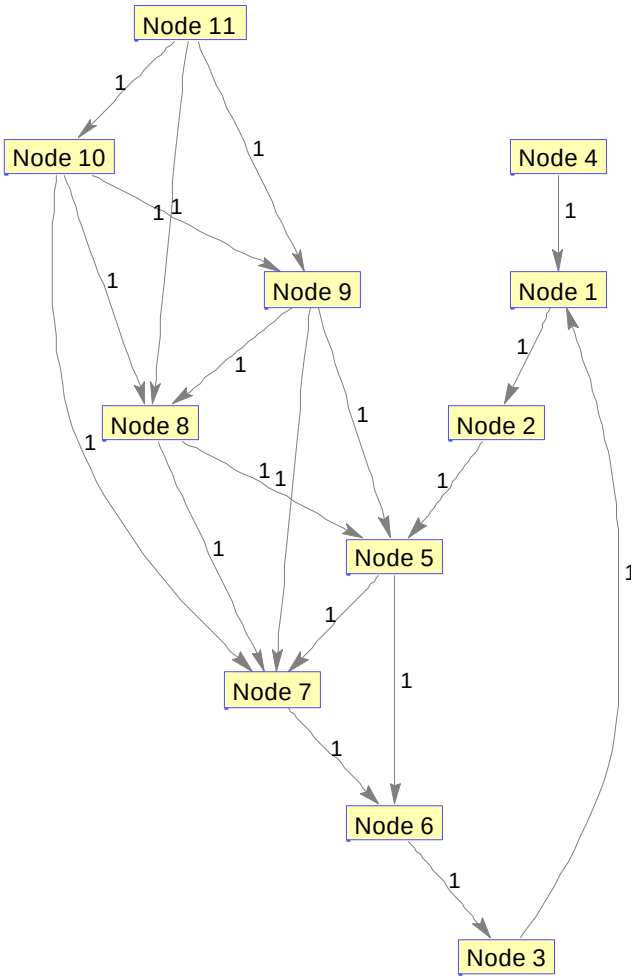


Figura 45: Resultados Abilene, topología BGP

Como vemos tenemos una topología iBGP con 19 sesiones (enlaces del grafo), con esta cantidad de sesiones conseguimos que la red opere como un full mesh, caso en el que tendríamos 11 x 10 = 110 sesiones, con lo que conseguimos ahorrar un gran número de ellas.

$$\frac{19}{110} = 0.173 \qquad \frac{110}{19} = 5.79$$

La reducción es menor que en el caso de la red GEANT (el número de nodos también es menor), pero seguimos teniendo en la topología full-mesh casi 6 veces más sesiones que en la topología de reflexión de rutas

Como el grafo es algo extenso vamos a mostrar la tabla con las sesiones, igual que hicimos en el caso anterior.

Origen	Destino		Origen	Destino		Origen	Destino		Origen	Destino
3	1		**2**	5		**5**	7		**9**	8
4	1		**8**	5		**8**	7		**10**	8
1	2		**9**	5		**9**	7		**11**	8
6	3		**5**	6		**10**	7		**10**	9
			7	6		**11**	10		**11**	9

Tabla 5: Sesiones de red Abilene

El peso IGP asociado a nuestra solución es: 16182.

El peso IGP asociado a una topología full mesh sobre Abilene: 214642.

4.3 Resultados obtenidos mediante nuestro algoritmo (greedy)

Vamos a probar nuestra implementación con las mismas redes utilizadas anteriormente para poder comparar los resultados. En estos resultados la ordenación utilizada es la misma que en el programa anterior, es decir no se ha variado la ordenación de los nodos. Posteriormente mostraremos soluciones con distintas ordenaciones. En estas pruebas no se ha impuesto ninguna restricción temporal a la función bintprog.

Red de 4 nodos con $D_{IGP} = \begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 2 & 1 \\ 3 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}$

```
Topologia final:

bgpUPfin =

     0     0     0     0
     1     0     0     0
     0     1     0     0
     0     1     1     0


bgpDOWNfin =

     0     1     0     0
     0     0     1     1
     0     0     0     1
     0     0     0     0

Elapsed time is 1.339066 seconds.
Biograph object with 4 nodes and 4 edges.
```

Figura 46: Resultados Greedy (4 nodos), Consola de Matlab

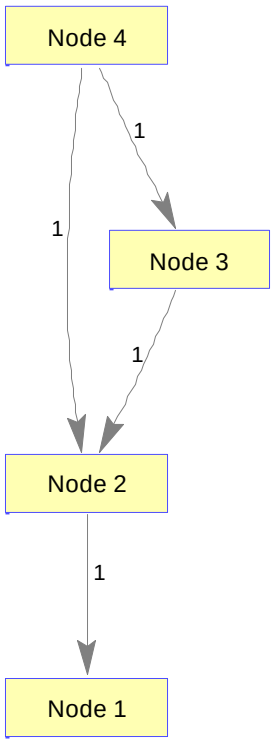


Figura 47: Resultados Greedy (4 nodos), topología BGP

Podemos ver que el resultado es bastante similar, también tiene además el mismo número de sesiones final (4). Para una red tan pequeña todavía no se aprecia que nuestro algoritmo es más rápido que el anterior (para más detalle ver resumen de resultados, sección 4.8).

La reducción con respecto al full mesh es igual que con el algoritmo utilizado anteriormente:

$$\frac{4}{4\ x\ 3} = \frac{4}{12} = \frac{1}{3}$$

Red de 5 nodos con $D_{IGP} = \begin{pmatrix} 0 & 4 & 5 & 2 & 5 \\ 4 & 0 & 8 & 3 & 6 \\ 5 & 8 & 0 & 7 & 2 \\ 2 & 3 & 7 & 0 & 7 \\ 5 & 6 & 2 & 7 & 0 \end{pmatrix}$

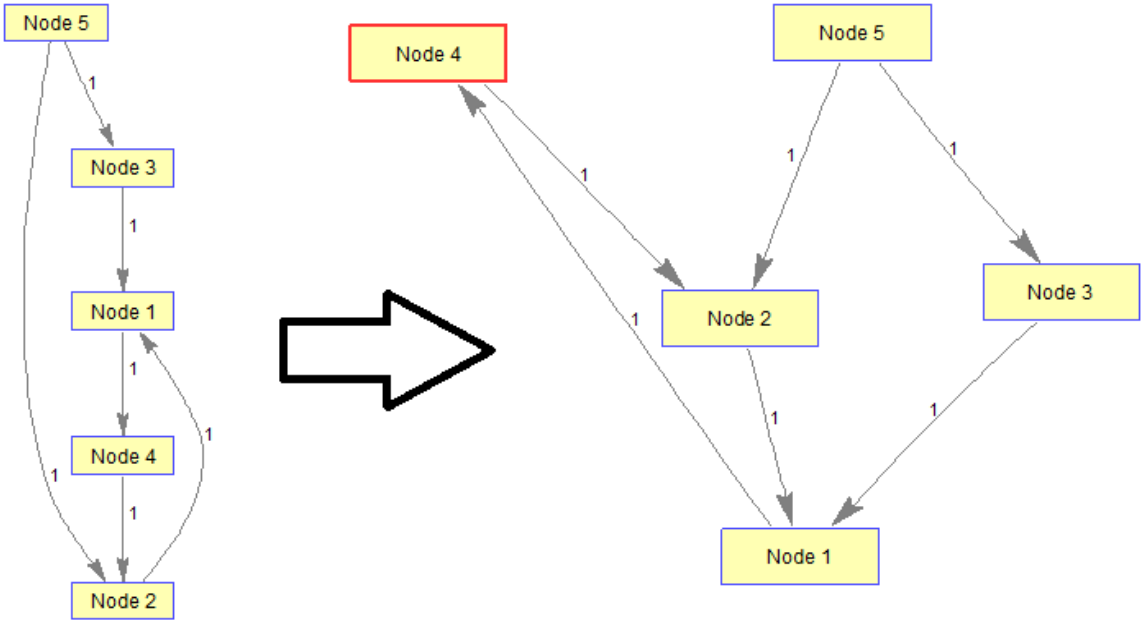


Figura 48: Resultados Greedy (5 nodos), topología BGP

```
bgpUPfin =

     0     0     0     1     0
     1     0     0     0     0
     1     0     0     0     0
     0     1     0     0     0
     0     1     1     0     0

bgpDOWNfin =

     0     1     1     0     0
     0     0     0     1     1
     0     0     0     0     1
     1     0     0     0     0
     0     0     0     0     0

Elapsed time is 1.510865 seconds.
Biograph object with 5 nodes and 6 edges.
```

Figura 49: Resultados Greedy (5 nodos), Consola de Matlab

Como vemos nos sale una topología un tanto extraña, pero al recolocar los nodos vemos que es muy parecida a la que obtenemos con el algoritmo BUOB. Obtenemos una topología con 6 sesiones iBGP instaladas. En un tiempo de 1.510865 segundos. Este tiempo ya es menor que el requerido con el algoritmo anterior. La reducción con respecto a full mesh es la misma que con el algoritmo utilizado antermente (1/3).

Coste IGP asociado=22.

Red de 6 nodos con $D_{IGP} = \begin{pmatrix} 0 & 6 & 6 & 4 & 5 & 5 \\ 6 & 0 & 4 & 4 & 4 & 7 \\ 6 & 4 & 0 & 8 & 8 & 11 \\ 4 & 4 & 8 & 0 & 3 & 3 \\ 5 & 4 & 8 & 3 & 0 & 4 \\ 5 & 7 & 11 & 3 & 4 & 0 \end{pmatrix}$

```
Topologia final:

bgpUPfin =

     0     0     0     0     0     0
     1     0     0     0     0     0
     1     1     0     0     0     0
     1     1     0     0     0     0
     0     1     0     1     0     0
     1     0     0     1     0     0


bgpDOWNfin =

     0     1     1     1     0     1
     0     0     1     1     1     0
     0     0     0     0     0     0
     0     0     0     0     1     1
     0     0     0     0     0     0
     0     0     0     0     0     0

Elapsed time is 1.502644 seconds.
Biograph object with 6 nodes and 9 edges.
```

Figura 51: Resultados Greedy (6 nodos), Consola de Matlab

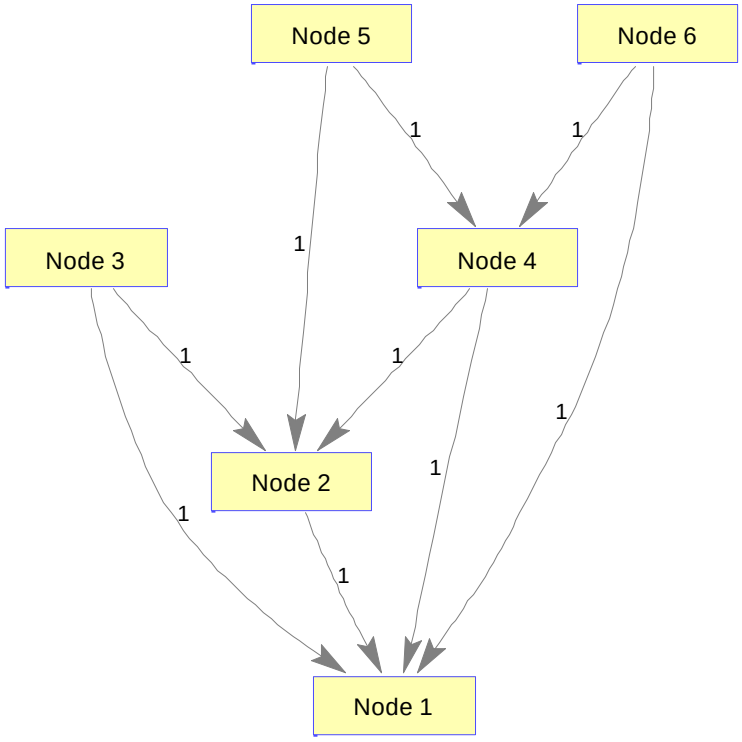


Figura 50: Resultados Greedy (6 nodos), topología BGP

El coste IGP asociado a esta topología es: 39.

Los resultados obtenidos son sorprendentes pues obtenemos menos sesiones que con el algoritmo de Buob (9 < 10). **Hemos obtenido ante una topología fm-óptima hallada en menos tiempo y con menos sesiones iBGP (también menos coste IGP asociado, 39 < 43) que con el programa desarrollado anteriormente.**

Reducción con respecto a full mesh: $\frac{9}{6\ x\ 5} = \frac{9}{30} = \frac{3}{10} = 0{,}3$

Esta reducción es mayor que con el algoritmo anterior: 0,3 < 0,33.

4.3.1 Greedy para red Abilene

Hemos probado nuestro programa con la red Abilene que probamos anteriormente, los resultados han sido los siguientes:

```
bgpUPfin =

     0     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0
     0     0     1     0     1     0     0     0     0     0     0
     0     0     0     0     1     1     0     0     0     0     0
     0     0     0     0     1     0     1     0     0     0     0
     0     0     0     0     1     0     1     1     0     0     0
     0     0     0     0     0     0     1     1     0     0     0
     0     0     0     0     0     0     0     1     0     1     0

bgpDOWNfin =

     0     1     1     1     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     1     1     0     0
     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     1     1     1     0
     0     0     0     0     0     0     0     0     1     1     1
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0     0

Elapsed time is 3.164842 seconds.
Biograph object with 11 nodes and 17 edges.
```

Figura 52: Resultados Greedy (Abilene), Consola de Matlab

Además de reducir el tiempo de ejecución de 16 a 3 segundos aproximadamente, tenemos una solución fm-optima con 2 sesiones menos que antes, la topología es bastante parecida, a continuación ponemos el grafo BGP donde sale representada

Reducción con respecto a full mesh:

$$\frac{17}{11\ x\ 10} = \frac{17}{110} = 0.154 \qquad \frac{\mathbf{11\ x\ 10}}{\mathbf{17}} = \frac{\mathbf{110}}{\mathbf{17}} = \mathbf{6.47}$$

Esta reducción es mayor que con el algoritmo anterior: 6.47 > 5.78.

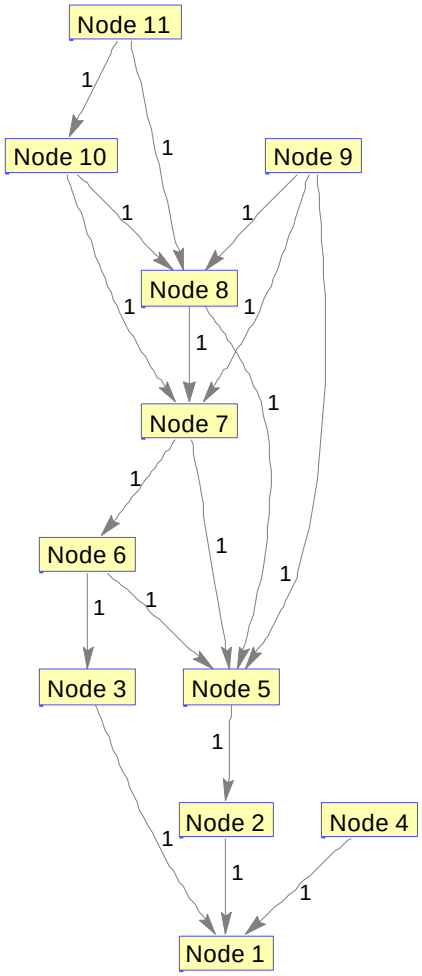


Figura 53: Resultados Greedy (Abilene), topología BGP

El coste IGP asociado a esta solución es: 14029, recordemos que el coste de full mesh es: 214642 y el de BUOB era 16182.

Ahora vamos a mostrar la tabla de sesiones instaladas. Las sesiones en rojo indican que son nuevas, es decir antes no existían, las sesiones en verde han cambiado de sentido:

Origen	Destino		Origen	Destino		Origen	Destino		Origen	Destino
2	1		**6**	5		**8**	7		**9**	8
3	1		**7**	5		**9**	7		**10**	8
4	1		**8**	5		**10**	7		**11**	8
5	2		**9**	5					**11**	10
6	3		**7**	6						

Tabla 6: Sesiones de red Abilene (Greedy)

4.3.2 Greedy para red GEANT

A continuación probamos con la red de GEANT, recordar que tiene 23 nodos, y con el programa anterior necesitábamos 74 enlaces para que funcionara como un full-mesh

```
Topologia final:
Elapsed time is 3.820340 seconds.
Biograph object with 23 nodes and 69 edges.
```

Figura 54: Resultados Greedy (GEANT), Consola de Matlab

El grafo no aporta demasiada información ya que con 23 nodos y 69 enlaces el resultado es poco visual, pero sí vemos que el tiempo de ejecución es mucho menor que en el programa anterior y además de esto tenemos 69 enlaces, es decir, 5 enlaces menos.

El resultado obtenido es muy bueno, en la literatura que hemos encontrado no se había conseguido alcanzar este número de sesiones en ningún caso, por ejemplo en nuestro artículo de referencia *"Designing optimal iBGP route-reflection topologies"*(**[1]**) muestran una solución con 74 enlaces para 22 nodos, nosotros con un nodo más hemos conseguido una red fm-optima con 69 enlaces.

Con el algoritmo greedy tenemos un coste IGP asociado de 184494 con 69 sesiones, con el de BUOB obtuvimos 198755 con 74 sesiones.

La reducción en el número de enlaces con respecto a full mesh es de:

$$\frac{69}{23\ x\ 22} = \frac{69}{504} = 0.137 \qquad \frac{\mathbf{23\ x\ 22}}{\mathbf{69}} = \frac{\mathbf{504}}{\mathbf{69}} = \mathbf{7.3}$$

Esta es una reducción mayor que la obtenida utilizando el algoritmo BUOB (7.3 > 6.83)

4.4 Resultados en redes grandes

Finalmente mostramos los resultados del algoritmo greedy en 2 redes de gran tamaño obtenidas a partir de la red tier-1 descrita en el apartado 4.7. Para obtener la topología de estas se han seleccionado los subconjuntos de nodos de esta red que están localizados geográficamente en Europa y América. De esta forma hemos conseguido dos redes nuevas para probar nuestro algoritmo greedy:

- La que llamaremos RedEuropa, que consta de 171 nodos.
- La RedAmérica, que está formada por 151 nodos.

Los resultados obtenidos para la RedEuropa son los siguientes:

- Tiempo de ejecución: Aproximadamente 8 horas en un portátil *(PROCESADOR: Intel® Core™ i3 CPU M33@2.13Ghz; MEM. RAM: 4 GB).*

En la iteración de mayor tamaño, al añadir el nodo 171, el programa debe comprobar si le llega ruta desde los otros 170 nodos (y si una ruta llega desde él hasta estos 170 nodos), en el caso de que no le llegue, buscara cuál es la forma menos costosa (en coste IGP) de hacer que le llegue la ruta correctamente (añadiendo sesiones claro está). Al final tendrá que solucionar para cada par (nodo, 171), una optimización en la que tendremos 680 (170 x 4 = 680) variables. Lo cual es algo ínfimo en comparación con las variables que usaríamos en cada iteración con el algoritmo replicado de Buob et al 2008 **[1]** (171 x 171 x 2 = 58482).

- Número de sesiones iBGP finales: **1033**.
- Coste total IGP asociado a esta solución**: 1.8705e+009**.
- Reducción del número de sesiones con respecto a full-mesh:

$$\frac{1033}{171\ x\ 170} = \frac{1033}{29070} = 0{,}035 \qquad \frac{171\ x\ 170}{1033} = \frac{29070}{1033} = 28.14$$

Como vemos es una reducción muy importante en el número de sesiones. Además esta solución sigue siendo mejorable variando la ordenación de los nodos (ver apartado 4.5.4).

Atendiendo al coste IGP asociado, con nuestra solución tenemos: 1.8705e+009 mientras que con full mesh: 1.8336e+011, es decir conseguimos una reducción de 2 órdenes de magnitud.

Al ser una red de 171 nodos los resultados gráficos no son suficientemente visibles por lo que no los incluimos en la memoria.

Los resultados para la RedAmérica son:

- Tiempo de ejecución: Aproximadamente 6 horas en un portátil *(PROCESADOR: Intel® Core™ i3 CPU M33@2.13Ghz; MEM. RAM: 4 GB).*

En la iteración de mayor tamaño, al añadir el nodo 151, el programa debe comprobar si le llega ruta desde los otros 150 nodos (y si una ruta llega desde él hasta estos 150 nodos), en el caso de que no le llegue, buscará cuál es la forma menos costosa (en coste IGP) de hacer que le llegue la ruta correctamente (añadiendo sesiones claro está). Al final tendrá que solucionar para cada par (nodo, 151), una optimización en la que tendremos 600 (150 x 4) variables. Lo cual es algo ínfimo en comparación con las variables que usaríamos en cada iteración con en el programa anterior (151 x 151 x 2 = 45602).

- Número de sesiones iBGP finales: **502**.
- Coste total IGP asociado a esta solución: **2.1368e+08**.
- Reducción del número de sesiones con respecto a full-mesh:

$$\frac{502}{151\ x\ 150} = \frac{502}{22650} = 0.022 \qquad \frac{151 * 150}{502} = \frac{22650}{502} = 45.12$$

Como vemos también tenemos una reducción muy importante (todavía mayor que la anterior) en el número de sesiones. Además esta solución sigue siendo mejorable si variamos la ordenación de los nodos (ver apartado 4.5.4).

Atendiendo al coste IGP asociado, con nuestra solución tenemos: 2.1368e+08 mientras que con full mesh: 2.1732e+010. Como en el caso anterior hemos obtenido una reducción de dos órdenes de magnitud.

Al igual que en el caso anterior, con 151 nodos los resultados gráficos no son suficientemente visibles por lo que no los incluimos en la memoria.

4.5 Algoritmo Greedy variando las ordenaciones de los nodos

A continuación mostramos los resultados obtenidos. En primer lugar probaremos con ordenaciones aleatorias, pudiendo obtener resultados mejores o peores que con la ordenación natural. Finalmente probaremos ordenaciones de acuerdo a criterios heurísticos.

4.5.1 Ordenaciones aleatorias en GEANT

Vamos a comenzar probando 50 ordenaciones aleatorias para la red de GEANT. Con la mejor ordenación conseguimos esta solución:

```
Biograph object with 23 nodes and 49 edges.

>> mejorBGPUP

mejorBGPUP =

     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0
     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     1     0     0     0     0     0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0
     1     1     0     0     0     0     0     0     1     0     0     0     0     0     1     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     1     0     0     0     0     0
```

Figura 55: Probando reordenación en GEANT, consola de Matlab

Una red fm-optima con únicamente 49 sesiones lo cual supone una mejora muy importante con respecto a lo que habíamos conseguido utilizando el orden natural de los nodos (74 sesiones).

De esta forma la reducción con respecto a full mesh es:

$$\frac{49}{23\ x\ 22} = \frac{49}{504} = 0.097 \qquad \frac{23 * 22}{49} = \frac{506}{49} = 10.3265$$

Tenemos 10 veces menos sesiones que en un full mesh y las rutas llegarán a todos los routers correctamente, la topología final que tenemos como resultado es la siguiente:

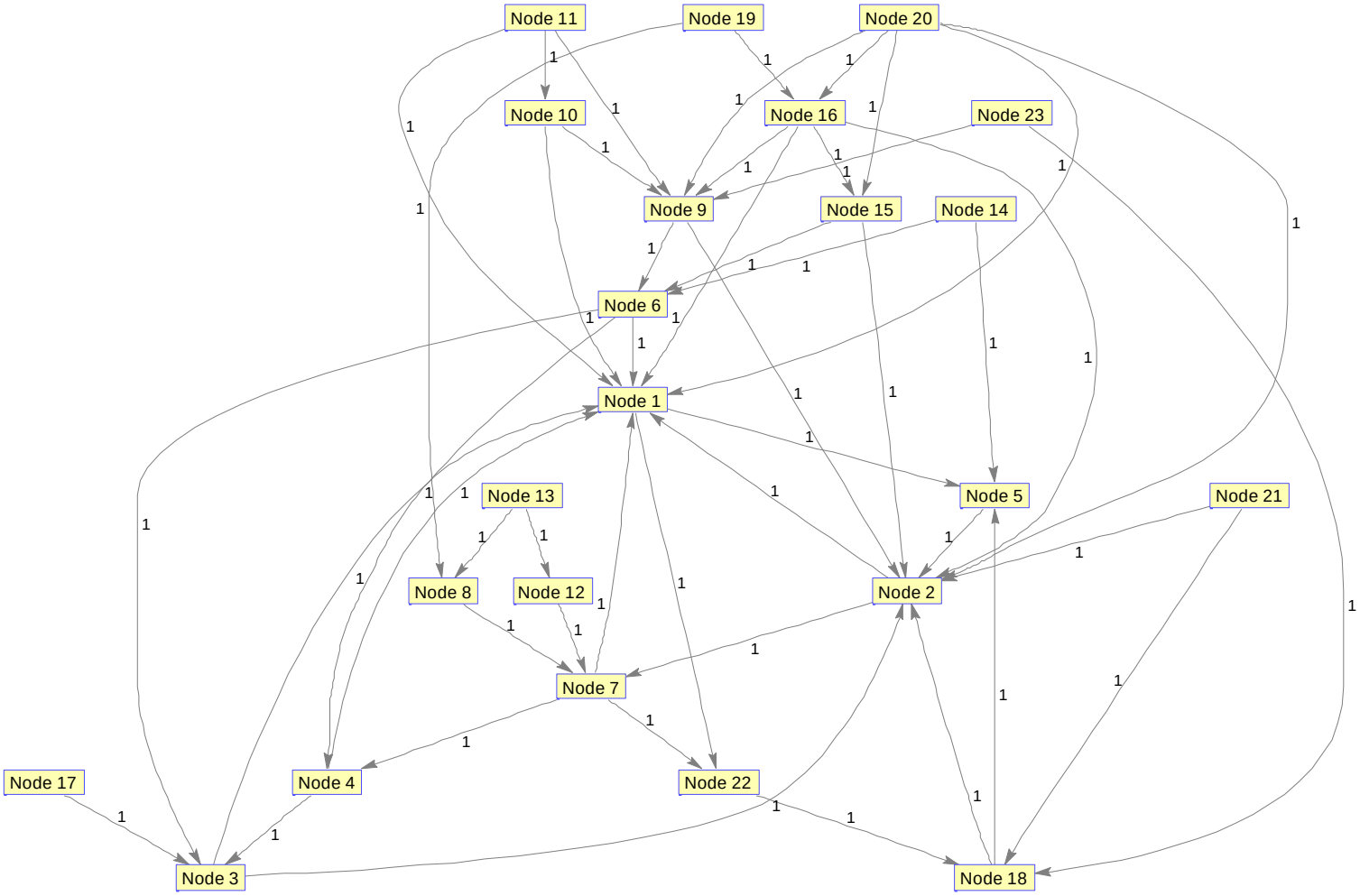


Figura 56: Probando reordenación en GEANT, Topología BGP

Con la mejor ordenación tenemos:

- 49 sesiones iBGP con un coste IGP asociado de: 78823.

Con la peor ordenación tenemos:

- 92 sesiones iBGP con un coste IGP asociado de: 299115.

Ahora recogemos los datos promedio de todas las ordenaciones probadas:

- (67.08 ± 9.0977) sesiones iBGP con un coste IGP asociado de: (151623±51198).

4.5.2 Ordenaciones aleatorias en Abilene

Para tener otro ejemplo vamos a probar también con la red de Abilene. Al ser de menor tamaño el tiempo es menor lo cual nos permite probar con más casos en esta ocasión probaremos con 100 ordenaciones diferentes. Con la mejor ordenación:

```
Topologia final:

mejorBGPUP =

     0     0     0     0     0     0     1     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0
     1     0     0     1     0     0     0     0     0     0     0
     1     0     0     0     1     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0
     1     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     1     0     0     0
     0     0     0     0     0     1     0     0     0     1     0
```

Figura 57: Probando reordenación en Abilene, consola de Matlab

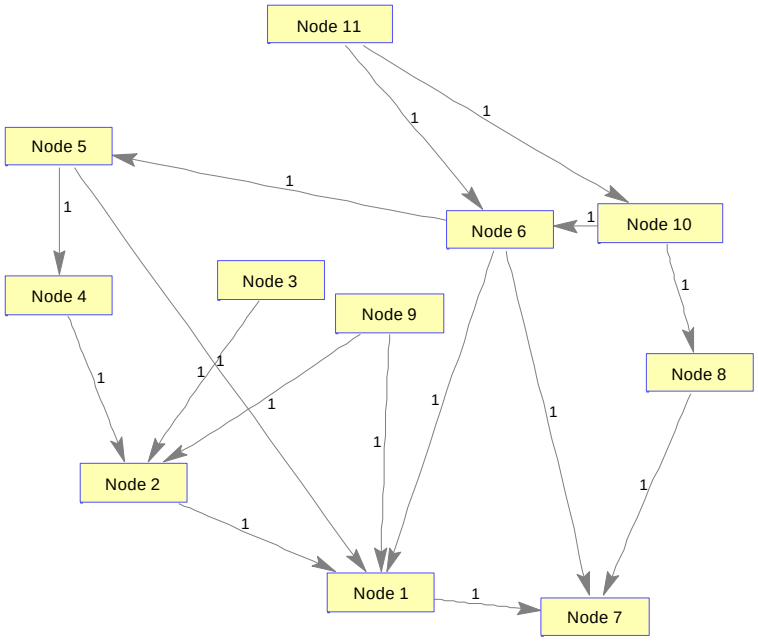


Figura 58: Probando reordenación en Abilene, Topología BGP

En esta ocasión tenemos 16 sesiones lo cual no es una mejora muy importante con respecto a lo que teníamos (17), creemos que esto se produce por las características IGP de la red.

Con la mejor ordenación tenemos:

- 16 sesiones iBGP con un coste IGP asociado de: 14127

Con la peor ordenación tenemos:

- 25 sesiones iBGP con un coste IGP asociado de: 38995

Ahora recogemos los datos promedio de todas las ordenaciones probadas:

- (20.42 ±2.0946) sesiones iBGP con un coste IGP asociado de. (24097 ± 4894.6)

4.5.3 Ordenaciones según criterios heurísticos

Ahora vamos a estudiar si las mejores ordenaciones tienen algún patrón común, lo cual sería muy beneficioso para utilizar el programa con redes más grandes, ya que ahorraríamos mucho tiempo si sabemos lo que debemos probar en lugar de probar ordenaciones al azar. Para estudiar las ordenaciones, vamos a tener en cuenta los siguientes parámetros (véase figura 59), coste total IGP de los enlaces que le llegan (columna 2), número de vecinos monohop (columna 3) y coste promedio de los enlaces del nodo (columna 4), la primera columna muestra el nodo.

```
>> estudioRed(geant, rpusada)

ans =
```

	Coste IGP	nº Enlaces	IGP (medio)
1	800	2	400
2	12470	2	6235
3	13300	2	6650
4	7130	6	1188.3
5	720	2	360
6	676	5	135.2
7	500	2	250
8	20800	5	4160
9	240	3	80
10	420	2	210
11	510	3	170
12	720	3	240
13	40050	2	20025
14	800	2	400
15	795	2	397.5
16	800	2	400
17	340	3	113.33
18	700	3	233.33
19	7235	4	1808.8
20	671	5	134.2
21	6865	3	2288.3
22	20995	6	3499.2
23	7205	5	1441

Figura 59: Tabla nodo → características

Vamos a probar a ordenar tanto de forma ascendente y descendente cada uno de estos parámetros y obtener los resultados con esa ordenación, de esta forma podremos intentar sacar alguna conclusión con respecto a las soluciones obtenidas en función de la ordenación.

Ordenando de menor a mayor el parámetro de coste IGP total obtenemos 74 sesiones con coste total asociado a su matriz de distancias IGP igual a 254856.

Ordenando de mayor a menor el parámetro de coste IGP total obtenemos 60 sesiones con coste total asociado a su matriz de distancias IGP igual a 87318, mucho mejor que la ordenación anterior.

Ahora ordenamos en orden ascendente por el numero de enlaces (en caso de empate orden ascendente de costeIGPtotal). Obtenemos 107 sesiones con coste total asociado a su matriz de distancias IGP igual 350350.

Cambiando el caso de empate a orden descendente de costeIGPtotal, obtenemos 127 sesiones con coste total asociado a su matriz de distancias IGP igual 492344.

Ordenando en orden descendente por el numero de enlaces (en caso de empate orden descendente de costeIGPtotal), obtenemos **43 sesiones** con coste total asociado a su matriz de distancias IGP igual **73163**.

Cambiando el caso de empate a orden ascendente de costeIGPtotal, obtenemos **42 sesiones** con coste total asociado a su matriz de distancias IGP igual **73063**.

Podemos observar que las 2 soluciones que obtenemos con orden descendente del número de enlaces, tienen un resultado muy bueno tanto en sesiones como en coste IGP asociado. El factor de desempate no parece ser relevante.

Finalmente probaremos ordenando por el coste promedio IGP:
- Ascendente: 58 sesiones con coste total IPG asociado igual 79831.
- Descendente: 71 sesiones con coste total IPG asociado igual 174014.

Como vemos parece interesante probar ordenando de forma descendente el número de enlaces, vamos a realizar las mismas pruebas para la red de Abilene para ver si podemos sacar conclusiones similares:

Ordenando en función del coste total IGP en Abilene:

- Ascendente: 16 sesiones con coste total IPG asociado igual 17609.
- Descendente: 19 sesiones con coste total IPG asociado igual 20323.

Ordenando en función del número de enlaces en Abilene:

- Ascendente: 22 sesiones con coste total IPG asociado igual 27682.
- Descendente: **16 sesiones** con coste total IPG asociado igual **14604**.

Ordenando en función del coste promedio en Abilene:

- Ascendente: 16 sesiones con coste total IPG asociado igual 16269.
- Descendente: 23 sesiones con coste total IPG asociado igual 27348.

La red Abilene es más pequeña por lo que las diferencias en las soluciones también se reducen pero si podemos ver que sigue siendo interesante ordenar los nodos por el número de enlaces de forma descendente, sobre todo si atendemos al coste IGP.

Las tablas siguientes recogen estos datos para facilitar la lectura de los mismos.

ORDENACIONES EN GEANT (23 NODOS)							
Parámetro	*Coste Total IGP*		*Numero de Enlaces*		*Coste Promedio IGP*		*Promedio*
Orden	Ascendente	Descendente	Ascendente	Descendente	Ascendente	Descendente	Aleatorio
Sesiones	74	60	118	**42**	58	71	68.06 ± 9.0719
Coste IGP	254856	87318	354631	**73063**	79831	174014	161630 ± 55714

Tabla 7: Ordenaciones en GEANT

ORDENACIONES EN ABILENE (11 NODOS)							
Parámetro	*Coste Total IGP*		*Numero de Enlaces*		*Coste Promedio IGP*		*Promedio*
Orden	Ascendente	Descendente	Ascendente	Descendente	Ascendente	Descendente	Aleatorio
Sesiones	16	19	22	**16**	16	23	20.42 ± 2.0946
Coste IGP	17609	20323	27682	**14604**	16269	27348	24097 ± 4894.6

Tabla 8: Ordenaciones en Abilene

La mejor ordenación es en ambos casos la del ***número de enlaces descendente***, esto parece lógico, ya que si ponemos al inicio los nodos con más enlaces directos es más probable que los nodos siguientes tengan enlace directo con ellos y por lo tanto su sesión BGP realizara menos saltos (uno si tienen enlace directo). Para entenderlo de forma más clara podemos ver la ordenación opuesta, si ponemos el primero a un nodo con solamente un enlace directo todos los demás nodos menos el nodo que está enlazado a él, si necesitan tener sesión BGP con el nodo inicial, esta costará siempre como mínimo el coste del enlace directo. A este coste habría que añadirle el resto de tramos IGP por los que pasaría la sesión antes de llegar al nodo inicial.

Para observar mejor estos resultados vamos a presentarlos en forma de histograma, de esta manera podremos saber cuan buena es nuestra ordenación escogida.

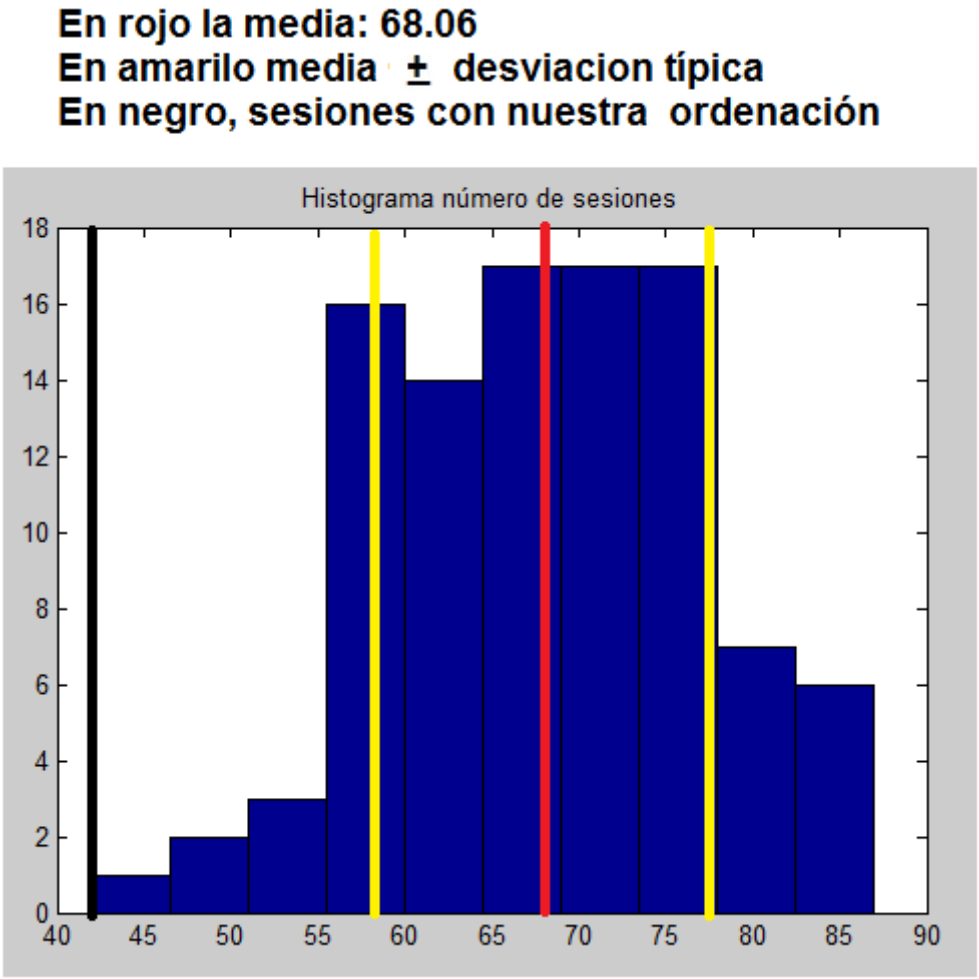


Figura 60: Histograma número de sesiones red GEANT

Como vemos en la figura 60 si utilizamos el orden descendente del número de enlaces obtenemos una solución mucho mejor que la gran mayoría de las soluciones aleatorias.

Tras comentar los resultados relacionados con el número de sesiones, vamos a presentar los histogramas del coste IGP asociado a esas sesiones.

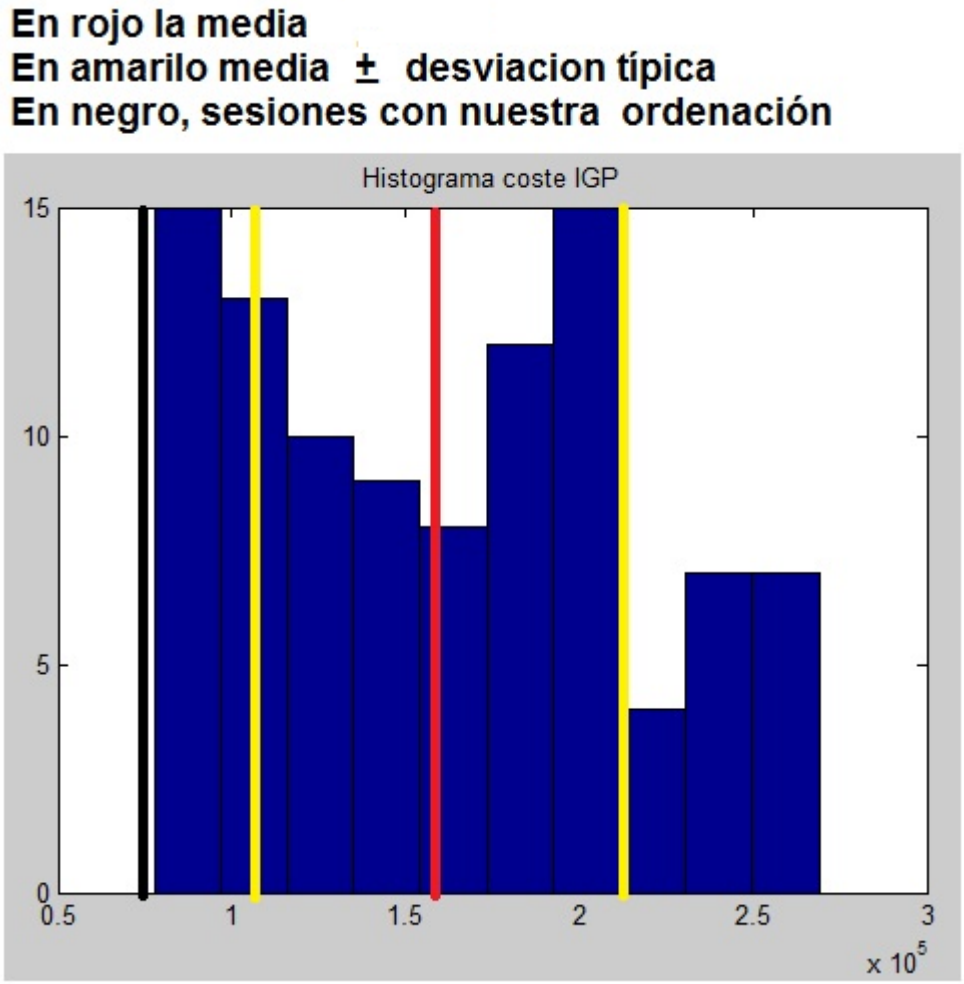


Figura 61: Histograma costes IGP red GEANT

Al igual que nos pasaba con el número de enlaces, utilizando la ordenación escogida obtenemos unas soluciones muy buenas si las comparamos con las obtenidas al utilizar ordenaciones aleatorias.

Tras comentar los resultados obtenidos en la red GEANT, vamos a ver si en la red Abilene sucede algo similar:

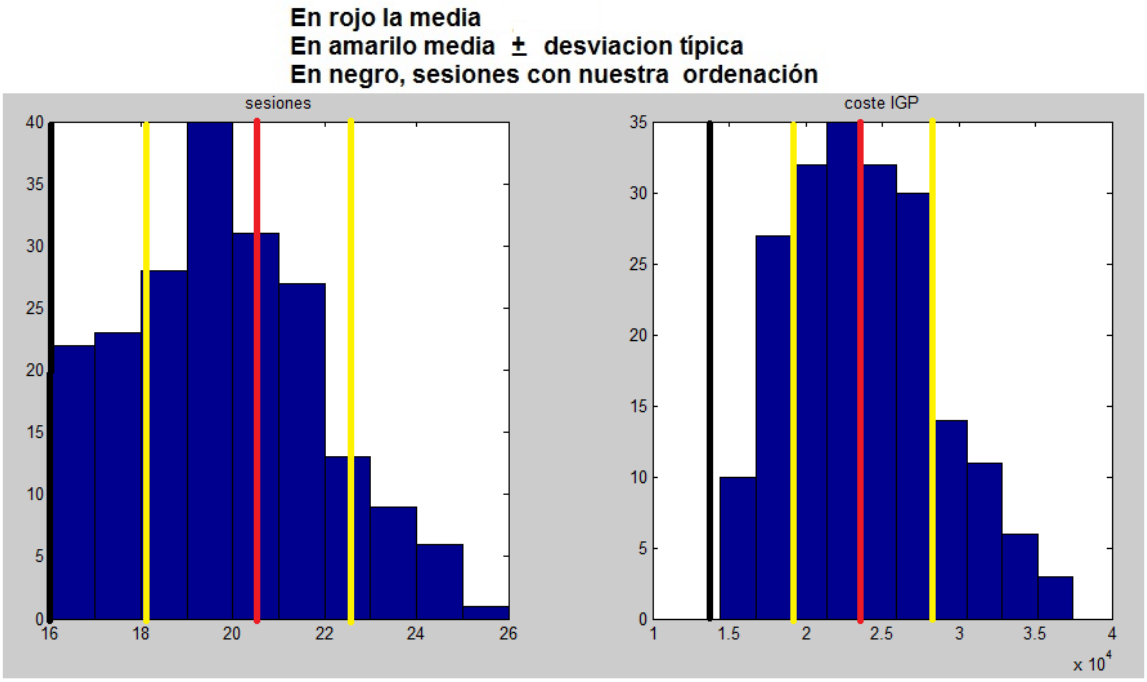


Figura 62: Histogramas de Abilene

Como vemos los resultados son similares a los de GEANT, nuestra solución es al menos igual que la mejor obtenida con las ordenaciones aleatorias (probando 200).

4.5.4 Resultados estadístico en redes de 30 nodos

Para comparar de una forma más fiable las distintas ordenaciones probadas anteriormente, vamos a probar a solucionar 200 redes de 30 nodos. Hemos creado estas 200 redes cogiendo sub-redes de la RedEuropa, seleccionando aleatoriamente 30 nodos de los 171 posibles. La única restricción que ponemos a la red es que sea conexa (que no haya distancia infinita IGP entre ningún par de nodos). Con esto pretendemos dar más generalidad a nuestros resultados.

Durante la ejecución hemos creado 200 redes de 30 nodos sobre las cuales hemos calculado: 200 topologías con nuestra implementación greedy, 200 topologías con la implementación extraida de Buob et al 2008 **[1]** y 200 topologías con greedy ordenado de forma descendente por el número de enlaces. La tabla 9 muestra los resultados obtenidos:

RESULTADOS REDES DE 30 NODOS ORDENACION DESCENDENTE DEL NUMERO DE ENLACES				
Programa	Tiempo Promedio	Número de sesiones promedio	Coste IGP promedio total/10^6	Coste IGP promedio total/CosteIGP(full mesh)
Greedy	47.856	114.515 ± 33.573	76982 ± 73807	0.0877 ± 0.0395
Buob	267.03	130.050 ± 32.321	85396 ± 79746	0.0980 ± 0.0395
Gr. ordenado	**46.135**	**110.420 ± 36.571**	**70355 ± 73052**	**0.0783 ± 0.0426**

Tabla 9: Resultados promedios en redes de 30 nodos

A la vista de los resultados podemos observar que al reordenar los nodos en orden descendente de número de enlaces, conseguimos reducir el número de sesiones establecidas. Además de esto, si atendemos al parámetro que verdaderamente estamos minimizando durante la ejecución, el coste IGP asociado a esas sesiones, podemos ver que ordenando los nodos, el coste IGP promedio obtenido también es menor que si cogemos los nodos de forma aleatoria.

En la tabla 10 mostramos los valores máximos y mínimos en el número de enlaces y en el coste IGP para cada implementación.

MÁXIMOS Y MÍNIMOS EN ENLACES Y COSTE IGP				
Programa	Min (nº enlaces)	Max (nº enlaces)	Min (coste IGP)	Max (coste IGP)
Greedy	50	225	951.1	4.1000e+005
Buob	60	232	1037	4.4800e+005
Gr. ordenado	**50**	**222**	**673.5**	**4.1000e+005**

Tabla 10: Máximos y mínimos en enlaces y coste IGP

A continuación vamos a mostrar un grafica en la que se puede observar la fuerte correlación que existe entre las distintas implementaciones.

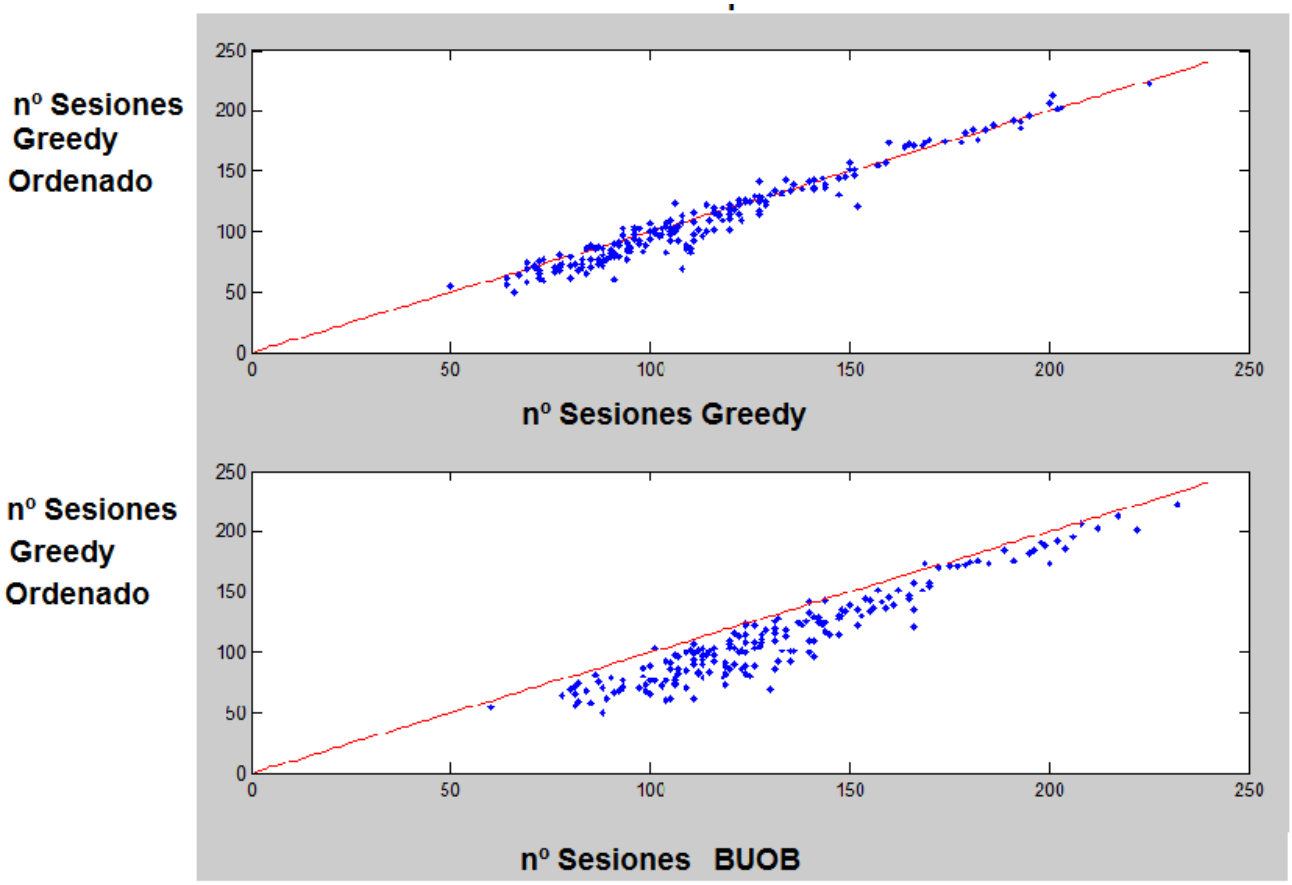


Figura 63: Correlación entre las implementaciones. (nº Sesiones)

A la vista de la figura 63, podemos observar que al comparar el greedy ordenado con cualquiera de las otras dos implementaciones, es más probable que obtengamos menos cantidad de sesiones con el greedy ordenado, que con BUOB o greedy, esto puede apreciarse en que en ambas gráficas hay mayor cantidad de puntos azules bajo la recta de pendiente igual a uno (recta roja en la gráfica) que por encima. Con el coste IGP la conclusión es la misma.

Con esta comprobación verificamos que es positivo utilizar este orden de nodos a la hora de realizar las topologías utilizando nuestro programa (greedy). Por esto vamos utilizar este criterio para ordenar los nodos en las redes de redEuropa, la redAmérica y la red global, los resultados de esta última se encuentran en el apartado 4.7.

Utilizando la ordenación estudiada (en orden descendente del número de enlaces) en la redEuropa conseguimos:

- Número de sesiones instaladas: **336**
- Coste IGP total asociado a estas sesiones:154625921
- Reducción en el numero de sesiones con respecto a full mesh: 336 / 29070 = **0.0116**
  Relación inversa: 29070 / 336 = **86.2069**
- Reducción en el coste IGP total asociado a las sesiones con respecto a full mesh:
  154625921 / 1.8336e+011 = 0.00084328
  Relación inversa: 1.8336e+011 / 154625921 = 1185.8
- Reducción en el numero de sesiones con respecto a ordenación natural:
  336 / 1033 = **0.325**
  Relación inversa: 1033 / 336 = **3.0744**
- Reducción en el coste IGP total asociado a las sesiones con respecto a ordenación natural: 154625921 / 1.8705e+009 = **0.0827**
  Relación inversa: 1.8705e+009 /154625921 = **12.0919**

Tenemos 86 veces menos sesiones que en full mesh, y 3 veces menos sesiones que con la ordenación natural de los nodos. El coste IGP asociado también se reduce de manera significativa.

Ordenando los nodos por su número de enlaces de forma descendente en la redAmerica conseguimos:

Número de sesiones instaladas: **304**

- Coste IGP total asociado a estas sesiones: **83408809**
- Reducción en el numero de sesiones con respecto a full mesh:
  304 / 22650 = **0.013422**;
  Relación inversa: 22650 / 304 = **74.507**
- Reducción en el coste IGP total asociado a las sesiones con respecto a full mesh:
  83408809 / 2.1732e+010 **= 0.0038381**;
  Relacion inversa: 2.1732e+010 / 83408809 = **260.55**
- Reducción en el numero de sesiones con respecto a ordenación natural:
  304 / 502 = **0.6056**
  Relación inversa: 502 / 304 = **1.6513**
- Reducción en el coste IGP total asociado a las sesiones con respecto a ordenación natural:
  83408809 / 2.1368e+08 = **0.3903**
  Relacion inversa: 2.1368e+08 / 83408809 = **2.568**

Tenemos 74 veces menos sesiones que en full mesh, y 1.6 veces menos sesiones que con la ordenación natural de los nodos. El coste IGP asociado también se reduce de manera significativa.

4.6 Resultados Añadiendo Redundancia

Este programa, en el cual introducimos el concepto de la redundancia lo probaremos únicamente en las redes GEANT, Abilene y en la tier-1 completa (los resultados de la tier-1 se encuentran en el apartado 4.7). Probar este programa en redes creadas por nosotros tiene poco sentido, ya que está orientado a su uso en redes reales. Como punto de partida para las topologías utilizaremos la solución obtenida a partir del algoritmo de Buob, ya que al tener (generalmente) más sesiones parte siendo más robusta que la nuestra, en cualquier caso si utilizáramos como punto inicial la nuestra, la solución final es muy parecida.

4.6.1 Redundancia en GEANT

Con el algoritmo de Buob conseguíamos una topología fm-optima con 74 enlaces, para hacerla robusta simulamos un fallo en cada nodo por separado obteniendo los siguientes resultados:

```
Topologia final:
Elapsed time is 110.569747 seconds.
Biograph object with 23 nodes and 146 edges.
```

Figura 64: Resultados Redundancia GEANT

Como vemos con 146 enlaces logramos una topología "robusta", entrecomillado porque no consideramos que pueden fallar simultáneamente 2 o más nodos, pero creemos que esta es una solución de compromiso.

En nuestro artículo de referencia (*Buob et al 2008* **[1])**, también consideran un apartado de robustez, pero lo enfocan de distinta manera, ellos siguen minimizando la función de coste IGP mientras que nosotros simplemente creamos una nueva sesión cada vez que por una avería se pierde la fm-optimalidad en un par de nodos. De esta manera mientras que su solución precisa de 172 sesiones nosotros utilizamos únicamente 146.

Con esta solución seguimos estando muy por debajo del número de sesiones requerido en un full mesh. $\frac{146}{23\ x\ 22} = \frac{146}{504} = 0.29$ $\frac{23 * 22}{146} = \frac{506}{146} = 3.4658$

4.6.2 Redundancia en Abilene

En esta red sí podemos mostrar los resultados de forma visual ya que al ser 10 nodos todavía puede apreciarse la topología:

```
Topologia final:

mUps =

     0     0     0     0     1     0     0     1     0     0     0
     1     0     1     1     0     1     0     1     1     0     0
     1     0     0     1     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0
     1     0     1     0     1     0     0     0     0     1     0
     0     0     1     0     1     1     0     0     0     0     1
     0     0     0     0     1     1     1     0     0     0     0
     1     0     0     0     1     0     1     1     0     0     0
     0     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     0     0     1     1     1     0

Elapsed time is 13.252480 seconds.
Biograph object with 11 nodes and 33 edges.
```

Figura 65: Resultados Redundancia (Abilene), consola de Matlab

Vemos que en únicamente 13 segundos hemos conseguido esta topología robusta para la red de Abilene, en esta ocasión necesitamos instalar 33 sesiones lo cual nos sigue situando muy por debajo del full mesh.

$$\frac{33}{11\ x\ 10} = \frac{33}{110} = 0.3 \qquad \frac{11 * 10}{33} = \frac{110}{33} = 3.33$$

Con un tercio de las sesiones conseguimos una topología fm-óptima que además es robusta frente a fallos aislados en los nodos. Esto supone un ahorro importante tanto en la memoria utilizada en las tablas de ruta como en el número de mensajes intercambiados entre los routers.

A continuación vamos a mostrar el grafo BGP que muestra la topología, en ella podemos apreciar, que se han añadido más sesiones BGP.

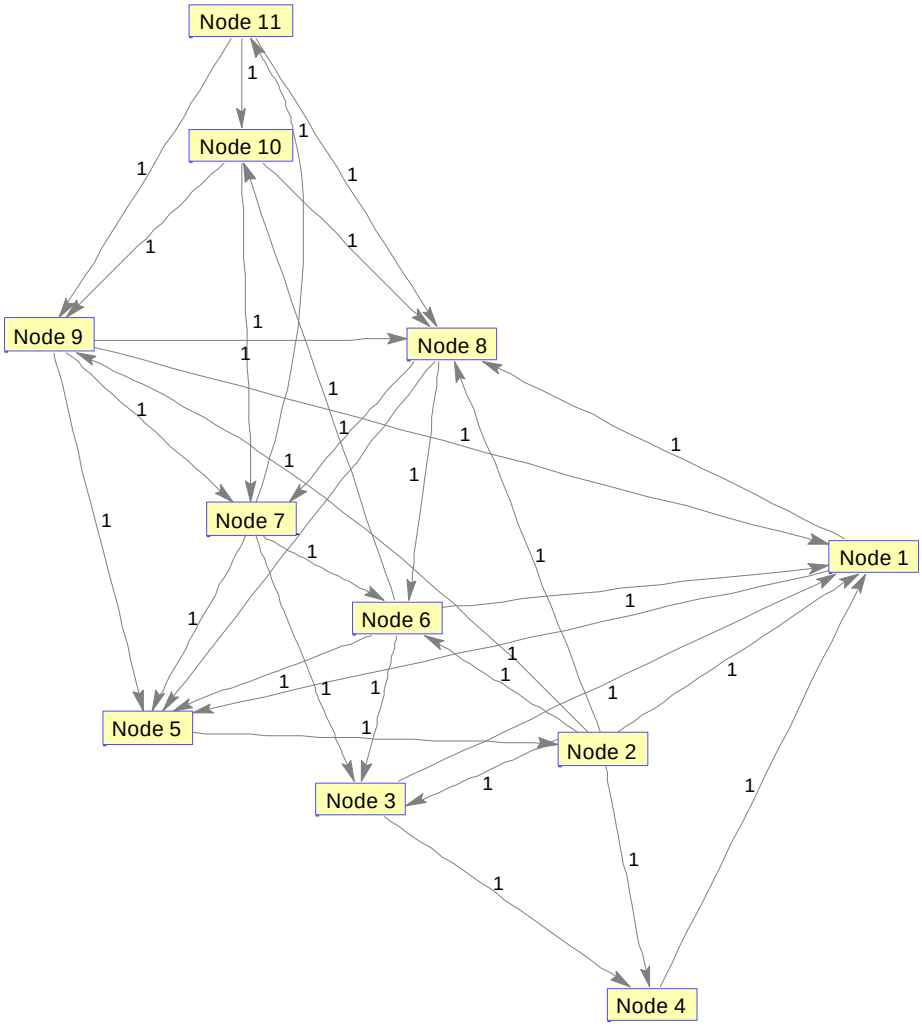


Figura 66: Resultados Redundancia (Abilene), topología BGP

4.7 Resultados específicos en red Tier-1

Finalmente probamos nuestros desarrollos con la red de tipo tier-1. Esta red está formada por dos subredes a las que llamamos redEuropa (171 nodos) y redAmerica (151 nodos). Ambas redes están conectadas a través de un número reducido de enlaces interoceánicos. El aspecto general de la red es muy similar al de la figura 67.

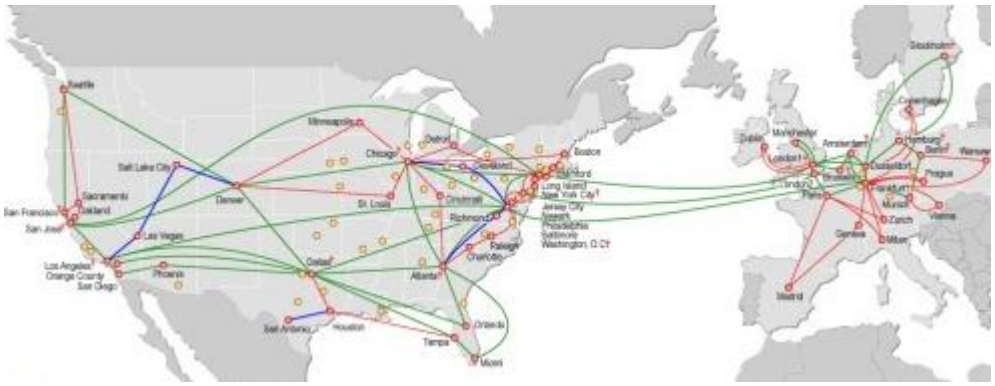


Figura 67: Distribución geográfica de la Red tier-1

Pensamos en hacer una topología BGP atendiendo a la disposición geográfica de los routers. De esta forma podemos observar que en los enlaces entre routers europeos y americanos son muy pocos (enlaces interoceánicos), por ello podíamos "dividir" la red en 2 subredes, la americana y la europea (redes utilizadas en los apartados anteriores). Siguiendo esta máxima se decidió establecer una topología full mesh en Europa, otra del mismo tipo en América y además de estas, otra topología full mesh entre los nodos con enlaces interoceánicos, de esta forma teníamos algo semejante a esto:

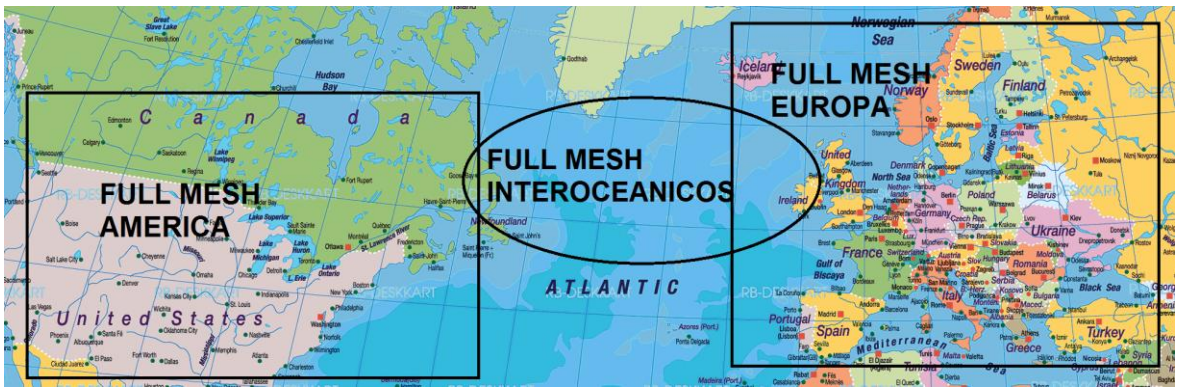


Figura 68: Topología orientativa resultante

Como era de esperar al utilizar la función comprobarRuta para cada par de nodos obteníamos una respuesta afirmativa en todos ellos, lo cual nos indica que estamos ante una topología fm-óptima. Aunque sabemos que el número de sesiones se podría reducir de forma considerable, con esta configuración tan simple ya hemos conseguido una reducción importante, ya que tenemos únicamente el siguiente número de sesiones:

```
>> sum(sum(BGPmatrix))

ans =

       51852
```

Con estas 51852 ya conseguimos una reducción importante con respecto al número de sesiones que tendríamos en un full mesh:

$$\frac{51852}{322\ x\ 321} = \frac{51852}{103362} = 0.5017 \qquad \frac{322 * 321}{51852} = \frac{103362}{51852} = 1.9934$$

Conseguimos que la red opere de forma óptima con prácticamente la mitad de las sesiones. Además de esto con esta red el máximo número de saltos entre cualquier par de routers en sentido BGP es de 3 lo cual nos da un retardo muy acotado y manejable

4.7.1 Greedy en Red Tier-1, 325 nodos

Al aplicar el algoritmo a la red de tipo tier-1 nos dimos cuenta de que no era tan rápido como nos pensábamos, encontrar la topología BGP adecuada para esta red llevó muchas horas de ejecución, concretamente unos 5 días de ejecución en un ordenador no demasiado potente (*PROCESADOR: Intel® Core™ i3 CPU M33@2.13Ghz; MEM. RAM: 4 GB*).

Los resultados obtenidos son los siguientes:

- Número total de sesiones iBGP instaladas: **1752**.
- Reducción de sesiones con respecto a full mesh:

$$\frac{1752}{325\,x\,324} = \frac{1752}{105300} = 0.0166 \qquad \frac{325 * 324}{1752} = \frac{105300}{1752} = 60.1027$$

Como vemos es una reducción de sesiones muy importante, aproximadamente tenemos 60 veces menos sesiones que antes lo cual nos supone un ahorro considerable en la memoria necesitada por los routers y también en el envío de mensajes con información de enrutado.

Al ser una red muy grande (325 nodos) se hace imposible mostrar los resultados de forma gráfica, pero sí podemos adelantar que nuestra topología no seguirá una estructura estrictamente jerárquica clásica, con lo que la implantación en el sistema real será bastante tediosa.

En esta topología no se ha incluido todavía redundancia, por lo que no podemos considerarla robusta ante averías.

4.7.1 Greedy con buena ordenación en Red Tier-1, 325 nodos

Vamos a probar nuestro algoritmo greedy con una buena ordenación sobre la red tier-1. Los resultados obtenidos son los siguientes:

Utilizando esta ordenación conseguimos:

- Número de sesiones instaladas: **730.**
- Coste IGP total asociado a estas sesiones: 283.595.535.
- Reducción en el numero de sesiones con respecto a full mesh:

$$\frac{730}{322\ x\ 321} = \frac{730}{103362} = 0.0071 \qquad \frac{322\ x\ 321}{730} = \frac{103362}{730} = 141.5918$$

- Reducción en el coste IGP total asociado a las sesiones con respecto a full mesh: **0.0020**;

  Relación inversa: 500.3367.
- Reducción en el numero de sesiones con respecto a ordenación natural:

  730/1700=0.4294;

  Relación inversa: 2.3288;

Como vemos conseguimos mejorar los resultados obtenidos al variar la ordenación consiguiendo una mejora ostensible tanto en el numero de sesiones como en el coste IGP asociado a las mismas.

4.7.2 Redundancia en tier-1

Ahora vamos aplicar la redundancia a la solución obtenida anteriormente para conseguir una red robusta ante averías.

- El número de sesiones necesarias para considerar la red como robusta es: **6196**

Con estas 6196 sesiones conseguimos una reducción con respecto al full mesh de:

$$\frac{6196}{325\ x\ 324} = \frac{6196}{105300} = 0.0588 \qquad \frac{325 * 324}{6196} = \frac{105300}{6196} = 16.99$$

Como vemos seguimos teniendo una reducción considerable en el número de sesiones, además nuestra red opera igual que un full mesh y es robusta frente a averías aisladas en los nodos.

4.8 Resumen de Resultados

Vamos a analizar los resultados obtenidos con programas desarrollados para cada una de las redes con las que hemos trabajado.

RESULTADOS PARA RED DE 4 NODOS		
Programa	BUOB	Greedy
Sesiones iBGP	4	4
Tiempo de ejecución (seg)	0.942048	1.339066
Restricciones	No	Greedy
Página	65	79

Tabla 11: Resultados red de 4 nodos

Para este caso el algoritmo greedy es más lento que el algoritmo BUOB pero esto sólo se debe a que el número de nodos es muy pequeño, observaremos en las siguientes tablas que esto no sucede.

A continuación mostramos los resultados obtenidos con la red de ejemplo de 5 nodos.

RESULTADOS PARA RED DE 5 NODOS		
Programa	BUOB	Greedy
Sesiones iBGP	6	6
Tiempo de ejecución (seg)	2.703863	1.361218
Restricciones	No	Greedy
Página	66	80

Tabla 12: Resultados red de 5 nodos

Con 5 nodos el greedy ya es más rápido que el algoritmo BUOB, y como vimos en el apartado de resultados específicos la solución es muy similar.

En la siguiente tabla mostramos los resultados obtenidos con una red de 6 nodos en la que al utilizar el nuestra implementación del algoritmo de Buob, coartamos su tiempo de ejecución, lo que mostraremos en una columna adicional.

RESULTADOS PARA RED DE 6 NODOS			
Programa	Buob	Buob	Greedy
Sesiones iBGP	10	10	9
Tiempo de ejecución (seg)	153.755207	4.722885	1.502644
Restricciones	No	Sí (temporal 1seg)	Greedy
Página	67	64	81

Tabla 13: Resultados red de 6 nodos

En esta red de 6 nodos ya podemos apreciar unas diferencias de tiempos importantes, y no solo eso, ya que con el Greedy tenemos una sesión menos lo cual es interesante mientras se mantenga la fm-optimalidad.

Hasta este momento hemos mostrado resultados obtenidos con redes creadas por nosotros de forma aleatoria, en adelante mostraremos resultados obtenidos con redes reales

RESULTADOS PARA RED GEANT 23NODOS			
Programa	BUOB	Greedy	Redundancia
Sesiones iBGP	74	69	146
Tiempo de ejecución (seg)	77.094347	3.791004	110.569747
Restricciones	SI (temporal 1 seg)	Greedy	SI (temporal 1 seg)
Coste IGP (distancias IGP)	198755	184494	No comparable
Robustez	No	No	Sí
Página	72	75	100

Tabla 14: Resultados red de GEANT

Estos son los resultados obtenidos para la red de GEANT como vemos nuestro algoritmo (Greedy) es más rápido y la solución final tiene menos sesiones iBGP y menos coste IGP (utilizando para su cálculo la matriz de distancias IGP). En cuanto a la redundancia la solución se toma partiendo de la obtenida con BUOB (para partir de una solución más robusta, mas sesiones suele implicar mayor robustez), no podemos comparar ni tiempo ni costes IGP puesto que está dando solución a "otro" problema, porque las soluciones anteriores no incluían robustez. Aun así podemos observar que el incremento temporal es pequeño con respecto al tiempo obtenido con BUOB.

Ahora mostraremos los resultados obtenidos en la red Abilene:

RESULTADOS PARA RED ABILENE 11 NODOS			
Programa	BUOB	Greedy	Redundancia
Sesiones iBGP	19	17	33
Tiempo de ejecución (seg)	11.078309	2.872844	12.830507
Restricciones	SI (temporal 1 seg)	Greedy	SI (temporal 1 seg)
Coste IGP (distancias IGP)	16182	14029	No comparable
Robustez	No	No	Sí
Página	84	82	101

Tabla 15: Resultados red de Abilene

Como vemos en todas las pruebas que hemos recogido aquí con nuestro algoritmo de tipo Greedy, conseguimos resultados mejores que los obtenidos con el algoritmo de BUOB y además en menos tiempo.

Además de esto, tenemos que añadir que todas las pruebas con el algoritmo greedy que hemos recogido en este resumen de los resultados, han sido realizadas sin variar la ordenación de los nodos, por lo que podemos decir que con una ordenación buena los resultados serían aun mejores.

Para escoger una buena ordenación, en caso de no ser posible probarlas todas (situación más común) hemos descubierto empíricamente que es recomendable probar ordenando los nodos según su número de enlaces directos IGP de forma descendente. Con esta ordenación hemos conseguido resultados mejores que con la ordenación natural en todos

los casos probados. Por lo que se recomienda su utilización en caso de usar nuestro algoritmo greedy.

A continuación amos a comentar explícitamente los resultados obtenidos en nuestra red tier-1:

RESULTADOS PARA RED TIER-1 325 NODOS				
Implementación	Unión de full mesh en América y Europa	Greedy	Greedy con buena ordenación	Greedy + Redundancia
Sesiones iBGP	51852	**1752**	**730**	**6196**
Tiempo de ejecución (días)	instantánea	5	5	5 + 5
Restricciones	NO	Greedy	Greedy	Greedy
Reducción frente a full mesh	1.9934	**60.1027**	**141.5918**	**16.99**
Robustez	Sí	No	No	Sí
Página	103	105	106	107

Como vemos con nuestro algoritmo (greedy) conseguimos construir una red fm-óptima con 60 veces menos sesiones que un full mesh, lo cual es un ahorro importantísimo tanto en memoria requerida por los routers como en cantidad de mensajes enviados.

Si utilizamos una buena ordenación en nuestro algoritmo los resultados son aun mejores, llegando a tener 140 veces menos sesiones instaladas que en un full mesh, lo cual es una mejora ostensible.

En la tabla podemos ver que esta solución no es robusta ante averías, por lo que queda descartada su implantación en la red real, sin embargo con nuestra solución con redundancia también obtenemos una reducción en el número de sesiones con respecto a full mesh (tenemos casi 17 veces menos sesiones instaladas en la red) y esta red sí podemos considerarla robusta. La implantación de esta topología en la red real sí podría llevarse a cabo, para ello únicamente habría que configurar cada uno de los nodos (trabajo bastante tedioso teniendo en cuenta que son 325 nodos) de acuerdo a los resultados obtenidos.

5

Conclusiones y trabajo futuro

5.1 Conclusiones

En este proyecto se han implementado varios algoritmos capaces de diseñar topologías BGP con reflexión de rutas a partir de la topología IGP del sistema autónomo. Las topologías diseñadas son óptimas en el sentido de que:

1. Propagan las rutas en el interior del sistema autónomo al igual que lo haría una topología de tipo full mesh (esta propiedad se llama fm-optimalidad).

2. Minimizan el coste IGP asociado a las sesiones iBGP establecidas.

Partiendo del algoritmo inicial de Buob et al. 2008 [1], se ha desarrollado un nuevo algoritmo que construye una topología BGP fm-óptima añadiendo los nodos de manera incremental. Este algoritmo encuentra soluciones de manera más rápida que el algoritmo original, pero las topologías resultantes, y el coste IGP asociado a las mismas, dependen en gran medida del orden en el que se añaden los nodos. No obstante es posible encontrar soluciones muy buenas eligiendo de manera conveniente la ordenación de los nodos.

En particular se observa que, si los nodos se van añadiendo por orden decreciente del número de enlaces IGP, las topologías obtenidas con el algoritmo incremental son claramente mejores que las reportadas en la literatura para el algoritmo original. Por tanto la combinación del algoritmo incremental y la ordenación apropiada de los nodos consigue un algoritmo que produce redes fm-óptimas mejores y en menos tiempo que los algoritmos de la literatura.

Finalmente se ha desarrollado una modificación del algoritmo que permite crear topologías fm-óptimas robustas frente al fallo de alguno de los nodos.

5.2 Trabajo futuro

El primer paso que se podría tomar tras el desarrollo de este proyecto sería tratar de simular el comportamiento de nuestras soluciones en redes reales, sabemos que nuestra red actúa como un full-mesh, pero sería conveniente ver como se mueve el tráfico, como se distribuyen las cargas de los enlaces etc. Esta simulación podría hacerse mediante una red de equipos con maquinas virtuales instaladas en ellos, estas maquinas virtuales simularían el comportamiento de cada router por separado, con lo que podríamos probar nuestras soluciones en una estructura muy similar a la red real. Por supuesto también tendríamos que simular las entradas de tráfico a la red y los destinos del mismo, esto podría hacerse de manera bastante fiel atendiendo a las estadísticas medias de la red real.

Cambiando un poco nuestra programación también podríamos minimizar directamente el número de sesiones iBGP instaladas en lugar de su coste IGP asociado para ello únicamente tendríamos que cambiar nuestra función a minimizar f, por una en la que el coste de todas las sesiones fuera igual.

Referencias

[1] M.O. Buob, S. Uhlig, M. Meulle: ***Designing optimal iBGP route-reflection topologies***. *2008.* In *IFIP Networking*, Singapore
http://perso.rd.francetelecom.fr/meulle/paper-ifipnetworking2008-buob-uhlig-meulle.pdf

[2] Y. Breitbart, M. Garofalakis, A. Gupta, A. Kumar, R. Rastogi: **On configuring BGP route reflectors**. 2007. Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference
http://www.softnet.tuc.gr/~minos/Papers/comsware07.pdf

[3] R. Dube: ***A comparison of scaling techniques for BGP***. *1999. SIGCOMM Computer Communication Review*, vol. 29, no. 3.
http://www.loria.fr/~ichris/Teaching/p44-dube.pdf

[4] J.G. Scudder, R. Dube: ***BGP scaling techniques revisited***. *1999.* SIGCOMM Comput. Commun.
http://www.loria.fr/~ichris/Teaching/p22-scudder.pdf

[5] L. Xiao, J. Wang, K. Nahrstedt: ***Optimizing iBGP route reflection network***. 2003. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 3, pages 1765–1769 vol.3
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.58.4686

[6] M.O. Buob, M. Meulle, S. Uhlig: ***Checking for optimal egress points in iBGP routing***. 2007. Proc. of the 6th IEEE International Workshop on the Design of Reliable Communication Networks (DRCN 2007),
http://marcolivier.buob.pagesperso-orange.fr/pdf/drcn07_buob_meulle_uhlig.pdf

[7] T. Bates: ***BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)*** 2006. RFC 2796
http://www.rfc-editor.org/rfc/rfc4456.txt

[8] S. Uhlig and S. Tandel: ***Quantifying the impact of route-reflection on BGP routes diversity inside a tier-1 network.*** 2006. SIGCOMM '09 Proceedings of the ACM SIGCOMM 2009 conference on Data communication
http://www.nas.ewi.tudelft.nl/people/Steve/papers/diversity.pdf

[9] M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan. ***How to construct a correct and scalable iBGP configuration***. 2006. In *IEEE INFOCOM*, Barcelona, Spain,
http://web.mit.edu/swastik/www/ibgp.pdf

[10] Página web de GEANT
http://www.geant.net/pages/home.aspx

[11] Página web de GEANT2
http://www.geant2.net/

[12] Página web de igen generator
http://informatique.umons.ac.be/networks/igen/

[13] A. Zinin. ***Analysis and minimization of microloops in link-state routing protocols***. 2005. Internet Engineering Task Force, Network Working Group Internet Draft
http://tools.ietf.org/html/draft-ietf-rtgwg-microloop-analysis-00

[14] Juniper. ***Selecting the best path***
http://www.juniper.net/techpubs/software/erx/erx50x/swconfig-routing-ol2/html/bgp-config10.html

[15] P. Traina, D. McPherson, and J. Scudder. ***Autonomous System Confederations for BGP.*** 2001. RFC 3065
http://www.ietf.org/rfc/rfc3065.txt

[16] M.O. Buob, M. Meulle, and J.L. Lutton. ***Un modèle de graphe et dedioïde pour le routage interdomaine.*** 2006. CFIP 06, pages 329–240
http://marcolivier.buob.pagesperso-orange.fr/pdf/cfip06_buob_meulle_lutton.pdf

[17] E. Rosen, A. Viswanathan and R. Callon. ***Multiprotocol Label Switching Architecture***. 2001. Internet Engineering Task Force, RFC3031
http://www.ietf.org/rfc/rfc3031.txt

[18] R. Dube and J.G. Scudder. ***Route reflection considered harmful***. 1999. Technical report.
http://members.tripod.com/~rohit_dube/papers/draft-dube-route-reflection-harmful-00.txt

[19] A. Rawat and M. A. Shayman. ***Preventing Persistent Oscillations and Loops in IBGP Configuration with Route Reflection.*** 2006, *Computer Networks*, pp. 3642–3665
http://portal.acm.org/citation.cfm?id=1228654

[20] Griffin, T., Wilfong, G.: ***On the Correctness of IBGP Configuration.*** 2002 In *Proc. of ACM SIGCOMM*
http://www.acm.org/sigcomm/sigcomm2002/papers/ibgp.pdf

Bibliografía

[1] BGP4 Inter-Domain Routing in the Internet, Stewart, John W. III . Boston [etc.]: Addison-Wesley, cop. 1999. **ISBN**: 0201379511

[2] Pattern recognition and machine learning, Christopher M. Bishop. Springer. 2006.

Glosario

AS Autonomous Sistem

ASN AS Number

BGP Border Gateway Protocol

eBGP exterior BGP

EGP Exterior Gateway Protocol

full mesh mallado complete o total

GGP Gateway to Gateway Protocol

iBGP interior BGP

IGP Interior Gateway Protocol

IP Internet Protocol

MPLS Multiprotocol Label Switching

Nexthop BGP Gateway, punto de salida del AS hacia un prefijo

RR Route Reflector

TCP Transmission Control Protocol

Anexo A

MANUAL DEL PROGRAMADOR

1. Programa Algoritmo replicado de Buob 2008

Algoritmo replicado de Buob 2008

```
clc;
close all;
clear all;
%%

% [spMTXIGP]= CreacionMatrizIGPaleatoria3GTW (numeroNodos);
% igpMatrix=zeros(numeroNodos,numeroNodos);
%
% for k=1:numeroNodos
%    for m=1:numeroNodos
%       if m ~= k
%          igpMatrix(k,m)=graphshortestpath(spMTXIGP,k,m);
%       end
%    end
% end
%igpMatrix
%pause;
%igpMatrix = [0 1 3 2; 1 0 2 1; 3 2 0 1; 2 1 1 0];
%igpMatrix =[ 0 4 5 2 5; 4 0 8  3 6; 5 8 0 7 2; 2 3 7 0 7; 5 6 2 7 0];

igpMatrix =[ 0 6 6 4 5 5; 6 0 4 4 4 7; 6 4 0 8 8 11; 4 4 8 0 3 3; 5 4 8 3 0 4; 5 7 11 3 4 0];

numeroNodos=6;
nodoscuadrado = numeroNodos*numeroNodos;
%limitación temporal
OPTIONS = optimset('MaxTime',1);


% Hay una variable up y una down por cada par de nodos, los coeficientes de
% la funcion a minimizar son las componentes de la matriz igp:
f = reshape(igpMatrix,numel(igpMatrix),1);
f = repmat(f,2,1);

% Las restricciones iniciales son:
k = 1;
A = zeros(nodoscuadrado,nodoscuadrado *2);
b = ones(nodoscuadrado,1);
for i=1:numeroNodos
   for j=1:numeroNodos

      up = zeros(numeroNodos,numeroNodos);
      down = zeros(numeroNodos,numeroNodos);
      up(i,j) = 1;
```

```
        down(i,j) = 1;
        A(k,:) = [reshape(up,numel(up),1); reshape(down,numel(down),1)]';
        k = k + 1;

    end
end


k = 1;
Aeq = zeros(nodoscuadrado,nodoscuadrado * 2);
beq = zeros(nodoscuadrado,1);


for i=1:numeroNodos
    for j=1:numeroNodos

        up = zeros(numeroNodos,numeroNodos);
        down = zeros(numeroNodos,numeroNodos);
        up(i,j) = 1;
        down(j,i) = -1;
        Aeq(k,:) = [reshape(up,numel(up),1); reshape(down,numel(down),1)]';

        k = k + 1;

    end
end

x = bintprog(f,A,b,sparse(Aeq),beq);
mUps = reshape(x(1:nodoscuadrado),numeroNodos,numeroNodos);

%%
% Construir mi propia funcion comprueba ruta, a la que le paso la matriz
% igp y los ups y downs (ademas del par n,r)...
% fprintf('Ruta 1 -> 2: %d\n',comprobarRutaLF(igpMatrix,x,1,2));
% fprintf('------------------------------------------------------------\n');
dist2=inf;
tic
iteraciones=0;
x3= zeros(1, nodoscuadrado*2);
for k=1:numeroNodos
    for m=1:numeroNodos
        if m ~= k

            if comprobarRutaLF(igpMatrix,x,k,m)==0
                % Caso k->m, aÃ±ado restriccion:
                up = zeros(numeroNodos,numeroNodos);
                down = zeros(numeroNodos,numeroNodos);
                k
                m
                %Para el par k-m busca todos los caminos posibles que lo conectan
                for i=1:numeroNodos
```

```
            for j=1:numeroNodos
                if j ~= i
                    if mUps (j,i)~= 1
                        if j<=max(m,k) && i<=max (m, k) % las demas no nos aportaran nada
util

                            UpAux=zeros(numeroNodos,numeroNodos);
                            UpAux(i,j)=1;
                            %DownAux=UpAux';
vectorx= [reshape(UpAux,numel(UpAux),1); reshape(UpAux',numel(UpAux'),1)];
                            x2=x+vectorx;
                            for g=1:numel (x2)
                                if x2(g)==2
                                    x2(g)=1;
                                end
                            end

                            iteraciones=iteraciones+1;

                            if comprobarRutaLF(igpMatrix,x2,k,m)==1
                                x3=x2;
                                up(i,j)=-1;
                                down(i,j) = -1;
                            end
                        end
                    end
                end
            end
            Aaux = [reshape(up,numel(up),1); reshape(down,numel(down),1)]';
            baux = -1;

            A = [A; Aaux];
            b = [b; baux];

            %tic
            x = bintprog(f,A,b,Aeq,beq, x3, OPTIONS);
            %toc
            mUps = reshape(x(1:nodoscuadrado),numeroNodos,numeroNodos);
        end
    end
end
end



fprintf('Topologia final:\n');
for k=1:numeroNodos
    for m=1:numeroNodos
        if m ~= k
            if comprobarRutaLF(igpMatrix,x,k,m)==0
```

```
            malasunto=1
        end
    end
  end
end

mUps = reshape(x(1:nodoscuadrado),numeroNodos,numeroNodos)
mDowns = reshape(x((nodoscuadrado+1):end),numeroNodos,numeroNodos)

% for k=1:numeroNodos
%     for m=1:numeroNodos
%         if m ~= k
%             if comprobarRutamio(full(spMTXIGP), mUps, k, m)==0
%                 malasuntof=1
%             end
%         end
%     end
% end


toc
GrafoBGP = biograph(sparse(mUps), [], 'ShowWeights','on')
view(GrafoBGP)
iteraciones
```

---

res = comprobarRutaLF(igpMatrix,x,n,r)

```
function res = comprobarRutaLF(igpMatrix,x,n,r)
%
% igpMatrix = matriz IGP
% x = vector de ups y downs
% n = next hop
% r = router
%

nNodes = size(igpMatrix,1);

% Lo primero es reformatear el vector de ups y downs a matriz de
% conexiones:
n2 = nNodes*nNodes;
mUps = reshape(x(1:n2),nNodes,nNodes);
mDowns = reshape(x((n2+1):end),nNodes,nNodes);

drn = igpMatrix(r,n);

ix = find(igpMatrix(r,:) > drn);

numN = length(ix);


% Busco los nodos que pueden ocultar la ruta:
for i=1:nNodes
    din = igpMatrix(i,n);
    for j=1:numN
        if din >= igpMatrix(i,ix(j))
            % Me cargo este nodo poniendo su up y su down a 0:
            mUps(i,:) = zeros(nNodes,1);
            mUps(:,i) = zeros(1,nNodes);
            mDowns(i,:) = zeros(nNodes,1);
            mDowns(:,i) = zeros(1,nNodes);
            %fprintf('Me cargo el %d\n',i);
            break;
        end
    end
end

%fprintf('Tras eliminar nodos:\n');
mUps;
mDowns;

% Ahora tengo que construir el grafo extendido y verificar si hay ruta:
mExt = [mUps eye(nNodes); zeros(nNodes) mDowns];
res = ~isinf(graphshortestpath(sparse(mExt),n,nNodes+r));
```

2. Programa Algoritmo Greedy

Algoritmo Greedy

```
clc;
close all;
clear all;
%igpMatrix = [0 1 3 2; 1 0 2 1; 3 2 0 1; 2 1 1 0];
%igpMatrix =[ 0 4 5 2 5; 4 0 8  3 6; 5 8 0 7 2; 2 3 7 0 7; 5 6 2 7 0];
%igpMatrix =[ 0 6 6 4 5 5; 6 0 4 4 4 7; 6 4 0 8 8 11; 4 4 8 0 3 3; 5 4 8
3 0 4; 5 7 11 3 4 0];

% load fgeant
% for k=1:23
%    for m=1:23
%       if m ~= k
%           igpMatrix(k,m)=graphshortestpath(geant,k,m);
%       end
%    end
% end

% load fabilene
% for k=1:11
%    for m=1:11
%       if m ~= k
%           igpMatrix(k,m)=graphshortestpath(abilene,k,m);
%       end
%    end
% end
%
% numeroNodos=5;
%
% [spMTXIGP]= CreacionMatrizIGPaleatoria3GTW (numeroNodos);
% igpMatrix=zeros(numeroNodos,numeroNodos);
%
% for k=1:numeroNodos
%    for m=1:numeroNodos
%       if m ~= k
%           igpMatrix(k,m)=graphshortestpath(spMTXIGP,k,m);
%       end
%    end
% end

%ORDENACION BUENA
load datosiniciales %%ordenacion igpMatrix
rp=ordenacion;
 for i=1:322
    for j=1:322
        igpMatrixAux(i,j) = igpMatrix(rp(i),rp(j));
    end
 end
 igpMatrix=igpMatrixAux;

% Hay una variable up y una down por cada par de nodos, los coeficientes
de
% la funcion a minimizar son las componentes de la matriz igp:
[nNodes,col]=size(igpMatrix);
n2=nNodes*nNodes;
```

```
OPTIONS = optimset('MaxTime',1);
bgpUPfin=zeros(nNodes,nNodes);
bgpDOWNfin=zeros(nNodes,nNodes);

bgpUP=zeros(nNodes,nNodes);
bgpDOWN=bgpUP';

for k=2:nNodes
    k
    %% FUNCION f
      f=igpMatrix([1:k-1],k);
      f = repmat(f,4,1);

    %% DESIGUALDAD

    indice = 1;
    A = zeros((k-1)*2, (k-1)*2*2);
    b = ones((k-1)*2,1);
    for i=1:k
        for j=1:k
            if j~=i
                if i==k || j==k
                    up = zeros(1,(k-1)*2);
                    down = zeros(1,(k-1)*2);
                    up(1,indice)=1;
                    down(1,indice)=1;
  A(indice,:) = [reshape(up,numel(up),1); reshape(down,numel(down),1)]';
                    indice = indice + 1;
                end
            end
        end
    end


    %% IGUALDAD
    Aeq= zeros((k-1)*2, (k-1)*2*2);
    beq = zeros((k-1)*2,1);
    indice=1;

      for i=1:k
        for j=1:k
            if j~=i
                if i==k || j==k

                    up = zeros(1,(k-1)*2);
                    down = zeros(1,(k-1)*2);
                    up(1,indice)=1;
                    down=fliplr(up);
                    d=find(down==1);
                    down(d)=-1;
   Aeq(indice,:) = [reshape(up,numel(up),1);
reshape(down,numel(down),1)]';
                    indice=indice+1;
                end
            end
        end
      end


    xaux=[reshape(bgpUPfin,numel(bgpUPfin),1);
reshape(bgpDOWNfin,numel(bgpDOWNfin),1)]';
```

```
    for i=1:k
        for j=1:k
            if j~=i
                if i==k || j==k
                    ruta=comprobarRutaLF(igpMatrix,xaux,i,j);
                    if ruta==0;
                        tic
                        up = zeros(1,(k-1)*2);
                        down = zeros(1,(k-1)*2);

%Para el par i-j busca todos los caminos posibles que lo conectan
                        for o=1:k
                            for p=1:k

                                if p ~= o
                                    if o==k || p==k
                                        UpAux=zeros(nNodes,nNodes);
                                        UpAux(o,p)=1;
 vectorx= [reshape(UpAux,numel(UpAux),1);
reshape(UpAux',numel(UpAux'),1)]';
                                        x2=xaux+vectorx;
                                        g=find(x2==2);
                                        x2(g)=1;
           if comprobarRutaLF(igpMatrix,x2,i,j)==1
                                            if p==k
                                                up(o)=-1;
                                            elseif o==k
                                                up((k-1)+p)=-1;
                                            end
                                        end
                                    end
                                end
                            end
   Aaux = [reshape(up,numel(up),1); reshape(down,numel(down),1)]';
                    baux = -1;
                    A = [A; Aaux];
                    b = [b; baux];

                    end
                end
             end
            end
        end
        xfin = bintprog(f,A,b,Aeq,beq, [],OPTIONS);
        upsfin=xfin(1:(k-1)*2);
        bgpUP=zeros(nNodes,nNodes);
        for q=1:length(upsfin)
            if q<=(length(upsfin)/2)
                if upsfin(q)==1;
                    bgpUP(q,k)=1;
                end
            else
                if upsfin(q)==1;
                    bgpUP(k,q-k+1)=1;
                end
            end
        end
        bgpDOWN=bgpUP';
        bgpUPfin=bgpUP+bgpUPfin;
        bgpDOWNfin=bgpDOWN+bgpDOWNfin;
```

```
    save fGO bgpUPfin bgpDOWNfin igpMatrix k
end


fprintf('Topologia final:\n');
% x= [reshape(bgpUPfin,numel(bgpUPfin),1);
reshape(bgpDOWNfin,numel(bgpDOWNfin),1)]';
% bgpUPfin
% bgpDOWNfin
% pause
% for k=1:nNodes
%     for m=1:nNodes
%         if m ~= k
%             if comprobarRutaLF(igpMatrix,x,k,m)==0
%                 k
%                 m
%                 malasunto=1
%             end
%         end
%     end
% end
% numeroNodos=23;
% for k=1:numeroNodos
%     for m=1:numeroNodos
%         if m ~= k
%             if comprobarRutamio(full(spMTXIGP), bgpUPfin, k, m)==0
%             %if comprobarRutamio(geant, bgpUPfin, k, m)==0
%                 k
%                 m
%                 malasuntof=1
%             end
%         end
%     end
% end
% toc
%
% GrafoBGP = biograph(sparse(bgpUPfin), [], 'ShowWeights','on')
% view(GrafoBGP)
```

3. Programa Redundancia

Programa de aplicación de Redundancia

```
clc;
close all;
clear all;
%% CARGAR SOLUCION SIN REDUNDANCIA mUps

for quitado=1:numeroNodos
AuxmUps=mUps;
   for in=1:numeroNodos
      AuxmUps(quitado,in)=0;
      AuxmUps(in,quitado)=0;
   End
 xquitado=[reshape(AuxmUps,numel(mUps),1);
reshape(AuxmUps',numel(AuxmUps'),1)];

   for k=1:numeroNodos
      for m=1:numeroNodos
         if m ~= k
            if m~=quitado && k~=quitado
              if comprobarRutaLF(igpMatrix,xquitado,k,m)==0

                 AuxmUps (k,m)=1;
    xquitado= [reshape(AuxmUps,numel(mUps),1); reshape(AuxmUps',numel(mUps'),1)];
                 mUps=mUps + AuxmUps;
                 for w=1:numeroNodos
                    for e=1:numeroNodos
                    if mUps(w,e)==2
                        mUps(w,e)=1;
                    end
                    end
                 end
              end
            end
         end
      end
   end
 end

x= [reshape(mUps,numel(mUps),1); reshape(mUps',numel(mUps'),1)];
mUps = reshape(x(1:nodoscuadrado),numeroNodos,numeroNodos)
%mDowns = reshape(x((nodoscuadrado+1):end),numeroNodos,numeroNodos)
toc
GrafoBGP = biograph(sparse(mUps), [], 'ShowWeights','on')
view(GrafoBGP)
```

4. Otras funciones utilizadas

[Correcto] = comprobarRuta2(MTXIGP, MTXBGP, router, GTW)

```
function [Correcto] = comprobarRuta2(MTXIGP, MTXBGP, router, GTW)
%COMPROBAR RUTA
%Router=R;
%GTW= N;
%RRs= se deben pasar los RRs para hacer la MTX extendida
%   Detailed explanation goes here
   [filas columnas]= size (MTXIGP);
   spMTXIGP=sparse(MTXIGP);
   k=1;
   AuxMTXBGP=MTXBGP;
   %Vemos la distancia de R a N
   [distrouter,pathrouter,predrouter] = graphshortestpath(spMTXIGP,router, GTW);
   Q(1)=0;

   %CREAMOS EL CONJUNTO Q
   for i=1:columnas
      if i ~= GTW
         [dist,path,pred] = graphshortestpath(spMTXIGP,router,i);
         if dist > distrouter
            Q(k)=i;
            k=k+1;
         end
      end
   end


   if Q(1)~=0 || length(Q)>1
   %Creamos W
   for i=1:columnas
      [dist,path,pred] = graphshortestpath(spMTXIGP,i,GTW);

      for j=1:length (Q)
         [distq,pathq,predq] = graphshortestpath(spMTXIGP,i,Q(j));

         %si esta mas cerca de alguno de Q que de N
         if distq < dist
            %borramos el nodo de MTXBGP
            for (z =1:columnas)
               AuxMTXBGP(i,z)=0;
            end
            for (z =1:filas)
               AuxMTXBGP(z,i)=0;
            end

         end
```

```
    end
  end
    %Construimos grafo extendido
    mExt = [AuxMTXBGP eye(filas); zeros(filas) AuxMTXBGP'];
    res = graphshortestpath(sparse(mExt),GTW,filas+router);

    if res~=inf
       Correcto = 1;
    else
       Correcto = 0;
     end

  else
    Correcto=1;
  end
```

[costeIGP] = costetotal(mtxIGP, mtxBGP)

```
function [costeIGP] = costetotal(mtxIGP, mtxBGP)
%COSTETOTAL Summary of this function goes here
%   Detailed explanation goes here
    coste=mtxIGP.*mtxBGP;
    costeIGP=sum(sum(coste));
end
```

[resultados] = EstudioRed(mtxIGP)

```
 function [resultados ] = EstudioRed(mtxIGP, rpusada)
%ESTUDIORED Summary of this function goes here
%   Detailed explanation goes here
% format short g
mtxIGP=full(mtxIGP);
nNodes=length(mtxIGP(:,1));
ordenacion=zeros(1,nNodes);
for i=1:nNodes
   ordenacion(i)=i;
end
resultados=zeros(nNodes,3);
%Resultados:1-->costeIGPtotal
%           2-->nEnlaces
%           3-->costePromedio
    for i=1:nNodes
        resultados(i,2)=sum(mtxIGP(:,i));
        a=find(mtxIGP(:,i)~=0);
        resultados(i,3)=numel(a);
        resultados(i,4)=resultados(i,2)/resultados(i,3);

    end
resultados(:,1)=ordenacion';
end
```

Anexo B

CARTA DE RECOMENDACIÓN: JUNIPER NETWORKS

July, 11th 2011

Att: Carlos Paris, Computer Engineering department of UAM

Dear Sir:

After reviewing your research, I would like to express my sincere congratulations for the job done here. Therefore, on behalf of Juniper, below you have an endorsement text to be used as appropriately.

> *The placement of BGP route reflectors in a big network is a complex decision. The reduction of BGP sessions should not be at the expense of suboptimal routing, high convergence times or redundancy loss. In this regard, some research groups have already designed mathematical algorithms to optimize a route reflector topology with at least one of these requirements.*
>
> *The Juniper Networks Professional Services team started a collaboration in 2010 with Universidad Autonoma de Madrid (UAM), in order to evaluate the applicability of existing algorithms to a specific Tier-1 service provider network. This is the topic of the B. Sc. Thesis untitled ³Study of the optimal placement of BGP route reflectors², written by Carlos Alberto París Murillo under the guidance of Professor Luis Fernando Lago from the Computer Engineering department of UAM. This thesis is integrated in a long term project to evolve the planning strategies required to achieve an optimal and robust route reflector topology, both from a general perspective and applied to a specific carrier network.*
>
> *This study provides some light in the applicability of the different existing algorithms and it is a valuable contribution to the Telecommunication Industry. Juniper encourages the Research centers to follow Carlos Alberto París Murillo and Professor Luis Fernando Lago example to create a better New Network.*

Sincerely,

David Noguer Bau

Head of Service Provider Marketing,

Europe, Middle-East and Africa

Juniper Networks

JUNIPER NETWORKS
1194 North Mathilda Ave.
Sunnyvale, CA 94089
o +1408 745 2000
f +1408 745 2100
www.juniper.net

Anexo C

PRESUPUESTO

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 1.000 €
- Material de oficina ..... 100 €
- Software ..... 200 €
- Total de ejecución material ..... 1.300 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 208 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 78€

4) Honorarios Proyecto

- 1200 horas a 15 € / hora ..... 18000 €
- 100 horas de trabajo del tutor (25€ / hora)…….....2500 €

5) Material fungible

- Gastos de impresión..... 200 €
- Encuadernación..... 20 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 22228 €

7) I.V.A. aplicable

- 18% Subtotal Presupuesto ..... 4001,04 €

8) Total presupuesto

- Total Presupuesto..... 26.229,04 €

Madrid, Julio 2011

El Ingeniero Jefe de Proyecto

Fdo.: Carlos Alberto Paris Murillo
Ingeniero Superior de Telecomunicación

Anexo D

PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de Estudio de la localización óptima de reflectores de rutas BGP. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma,

por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.