UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN SISTEMAS DE TELECOMUNICACIÓN

Implementación de un simulador de redes óptica elásticas *multicore* en OMNeT++

Autor:

D. Iván Viloria Vázquez

Tutor:

Dr. Ramón J. Durán Barroso

Dr. Ignacio de Miguel Jiménez

Valladolid, 15 de Julio de 2020

TÍTULO:	Implementación de un simulador de redes óptica elásticas *multicore* en OMNeT++
AUTOR:	D. Iván Viloria Vázquez
TUTOR:	Dr. Ramón J. Durán Barroso
	Dr. Ignacio de Miguel Jiménez
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Dr. Ignacio de Miguel Jiménez
VOCAL:	Dr. Ramón J. Durán Barroso
SECRETARIO	Dra. Noemí Merayo Álvarez
FECHA:	15 de Julio de 2020
CALIFICACIÓN:	

Resumen

Las redes ópticas elásticas están basadas en conmutación de circuitos (*lightpaths*) y surgieron como evolución de las redes con encaminamiento por longitud de onda (WRON) para conseguir aprovechar el ancho de banda de las fibras ópticas. En un siguiente paso de esta evolución, se han desarrollado fibras *multicore* en las que varios núcleos se incluyen dentro de la estructura de la fibra óptica y con ello se aumenta la capacidad de las mismas utilizando multiplexación por división espacial (*Space Division Multiplexing*, SDM). Para establecer cada *lightpath* en estas redes es necesario solucionar el problema del encaminamiento y la asignación de un *core* y de espectro (*Routing, Core and Spectrum Allocation*, RCSA). El Grupo de Comunicaciones Ópticas (GCO) de la Universidad de Valladolid (UVa) tiene actualmente implementado un simulador de redes ópticas elásticas que no permite explotar SDM en caso de redes con fibras *multicore*. El objetivo de este Trabajo de Fin de Grado es adaptar el simulador para poder trabajar de forma eficiente en redes de este tipo y proponer nuevas versiones de los métodos propuestos por el grupo para el establecimiento de *lightpaths* que puedan ser utilizadas en redes equipadas con fibras *multicore*. La eficacia de estos métodos se ha comprobado en el simulador desarrollado.

Palabras clave

Redes ópticas, fibra *multicore*, Space Division Multiplexing, redes ópticas elásticas, redes WRON, OMNeT++, algoritmo RSA.

Abstract

Elastic optical networks are based on circuit switching (*lightpaths*) and they were developed as the evolution of wavelength-routed optical network (WRON) in order to exploit the huge bandwidth of optical fibres. In the next step on this evolution, multicore fibers have been developed where several cores are included inside the optic fibre structure in order to increases their capacity using space division multiplexing (SDM). When establishing a *lightpath* in these networks is necessary to solve the routing, core and spectrum allocation problem (RCSA). The Optical Communications Group of Universidad de Valladolid (UVa) has implemented an elastic optical network simulator which doesn't allow using SDM with multicore fiber networks. The objective of this Final Degree Project is to adapt that simulator to be able to efficiently work with this kind of networks and to propose new versions of the algorithms for lightpath establishment proposed by the GCO which can work in networks equipped with multicore fibers. The performance of these methods has been evaluated in the developed simulator.

Keywords

Optical networks, multicore fibers, Space Division Multiplexing, elastic optic networks, WRON networks, OMNeT++, RSA algorithm.

Agradecimientos

La investigación desarrollada en este Trabajo Fin de Grado ha sido financiada por el Ministerio de Ciencia, Innovación y Universidades en el marco del proyecto ONOFRE-2 (TEC2017-84423-C3-1- P) y la red de investigación Go 2Edge (RED2018-102585-T) y por el Fondo Europeo de Desarrollo Regional FEDER a través del proyecto DISRUPTIVE del Programa Interreg V -A España-Portugal (POCTEP) 2014-2020. Las opiniones son de exclusiva responsabilidad del autor que las emite.

Índice

Índice de figuras

Capítulo 1. Introducción

Desde la aparición de las primeras fibras ópticas diseñadas para comunicación, las redes basadas en este medio de transmisión, redes ópticas, han tenido un desarrollo enorme para conseguir aprovechar de forma eficiente el gran ancho de banda que proporciona este medio de transmisión. Las ventajas que presentan las redes ópticas respecto a las que utilizan otros medios de transmisión son muy numerosas: ofrece un mejor ancho de banda, inmunidad frente a las interferencias electromagnéticas, bajo coste de los equipos, gran durabilidad de los mismos y poco espacio necesario para albergarlos [2, 3].

Como hitos importantes en dicha evolución podemos señalar los siguientes. A finales de la década de 1970 y principios de la década de 1980, el refinamiento de los cables ópticos y el desarrollo de fuentes emisoras de luz y detectores de alta calidad y a bajo coste, impulsó el desarrollo de los sistemas de comunicaciones ópticas. En 1983 en Estados Unidos se implementó la primera Red Nacional de Fibra Óptica, a finales de los 80s se comenzaba a operar con mayores longitudes de onda [1]. A comienzo de los 90s se produce otro salto en las redes de fibra óptica al aparecer los amplificadores de fibra óptica dopada con erbio (EDFA), esta tecnología supero las limitaciones de la velocidad y permitía la amplifiación de señales en varias amplitudes de onda simultáneamente, esto último facilitó que se pudiera combinar con multiplexación por longitud de onda (*Wavelength Division Multiplexing*, WDM) [2].

La utilización de WDM suscitó la utilización de redes en las que la comunicación fuera puramente óptica, es decir no es necesario convertir las señales ópticas en electrónicas en los nodos intermedios para su procesamiento. Utilizando circuitos ópticos entre el nodo origen y el nodo destino, denominados *lightpaths*, conseguimos que la capa óptica de la red cuente con funciones de encaminamiento y conmutación. Las redes WDM que hacen uso de lightpaths se denominan WRONs (*Wavelength-Routed Optical Network*). En una evolución de las redes WRON aparecieron las redes ópticas elásticas o flexibles, las cuales mediante el uso de formatos de modulación como OFDM y Nyquist-WDM, permiten el establecimiento de supercanales y con ello aprovechar de forma más eficiente el espectro de las fibras. Actualmente el Grupo de Comunicaciones Ópticas de Valladolid tiene desarrollado un simulador de este tipo de redes que permite un aprovisionamiento flexible de recursos utilizando tecnologías elásticas. Dicho simulador está desarrollado en OMNeT++ [4, 5].

El desarrollo de fibras *multicore* y su uso en redes ópticas elásticas va a posibilitar el incremento del ancho de banda de estas redes utilizando multiplexación espacial (*Space Division Multiplexing*, SDM).

El objetivo de este trabajo fin de grado (TFG) es adaptar el simulador actual de redes ópticas elásticas del GCO para poder trabajar con redes que utilicen fibras *multicore* y por lo tanto SDM y WDM. Además de la validación del mismo, se adaptarán los métodos de asignación de espectro previamente desarrollados por el grupo para trabajar en estas redes y se realizará un estudio de rendimiento de cada uno de los métodos en

este nuevo entorno. En concreto, nos centraremos en los algoritmos *Fixed-Grid* que dividen el espectro en una rejilla y dentro de estos analizaremos los algoritmos *Fixed-Grid Joint* (que entrega el ancho de banda requerido de forma compacta) y *Fixed Grid Disjoint* (en el que se permite el establecimiento de *sublightpaths* y el uso de *traffic grooming* para proporcionar el ancho de banda requerido). Denominamos *sublightpaths* a cada una de las conexiones ópticas extremo a extremo que se establecen para satisfacer la demanda de un usuario en el caso que se pueda dividir dicha demanda y utilizar *traffic grooming* para proporcionar el ancho de banda demandado por el usuario. En concreto, las modificaciones que se han realizado al simulador de redes elásticas del GCO han sido las siguientes:

1. Implementación automática de una red en anillo indicando el número de nodos y el número de núcleos de la fibra *multicore* utilizada.
2. Adaptación de la topología NSFNet utilizando fibras *multicore*.
3. En el simulador actual, a la hora de resolver el problema del encaminamiento se utiliza el método *k-shortest paths* pero solo era posible utilizar el número de saltos como métrica. Se ha modificado el simulador para que se pudiera utilizar también el retardo de propagación como métrica. Ahora el usuario puede elegir entre estas dos métricas.
4. El simulador de redes elásticas utilizaba como parámetro de rendimiento la probabilidad de bloqueo. Se han implementado mejoras para que pueda estimar también el retardo de propagación medio y el número medio de *sublightpaths* que se generan.
5. Adapatación de los métodos de encaminamiento y asignación de espectro (RSA, *Routing and Spectrum Allocation*) implementados para ser utilizado en redes que utilizan SDM con fibras *multicore*.
6. Propuesta de dos metodologías RCSA (*Routing, Core and Spectrum Allocation*), basadas en las propuestas previas de métods RSA por el GCO, para la solución del problema de establecimiento de *lightpaths* en redes que utilizan SDM.

Finalmente se ha realizado un estudio mediante simulación para ver el rendimiento de dichos métodos con distintas configuraciones de la red y teniendo en cuenta distintas métricas: probabilidad de bloqueo (probabilidad que una conexión no pueda ser establecida por falta de recursos), retardo de propagación medio, número medio de *sublightpaths* a establecer por cada petición (en el caso de métodos *disjoint*) y el tiempo de cálculo que cada método necesita para obtener sus resultados.

Capítulo 2. Estado del arte

Las redes WRON funcionan completamente en el dominio óptico estableciendo circuitos ópticos extremo a extremo. Dichos circuitos ópticos se denominan *lightpaths*. Si la red no está equipada con convertidores en longitud de onda, un *lightpath* debe utilizar la misma proporción del espectro a lo largo de todo su camino. Los nodos de acceso tienen una serie transmisores y receptores sintonizables en longitud de onda y pueden utilizar distintos formatos de modulación. Un *lightpath* comienza en uno de los nodos de acceso utilizando uno de sus transmisores sintonizados en una longitud de onda concreta. Después habrá atravesará un conjunto de conmutadores ópticos (OXCs) que encaminaran este *lightpath* hasta el nodo de acceso del destino. En el nodo destino, un receptor convertirá la señal óptica en una señal eléctrica. Los distintos *lightpaths* pueden pasar por los mismos enlaces siempre que no utilicen la misma porción del espectro para que no haya interferencias entre ellos [3].

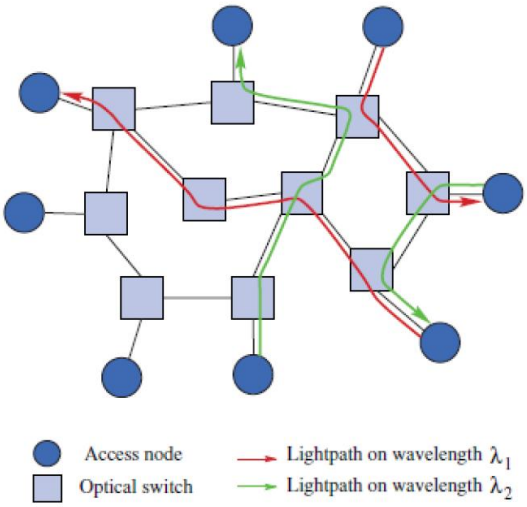

Figura 1. Ejemplo de red WRON

En las redes WRON dinámicas, las peticiones para el establecimiento de conexiones por parte de los usuarios llegan en tiempo real. Cuando llega una petición, el plano de control de estas redes debe resolver el problema RWA (*Routing and Wavelength Assignment*). Este problema se basa en asignar las longitudes de onda y las rutas a los *lightpaths* de la forma más eficiente posible, ya que los recursos son limitados y si no es posible asignar recursos a las peticiones estas serán rechazadas o bloqueadas y con ello aumentará la probabilidad de bloqueo de la red [3].

Para aumentar la eficiencia de las redes WRON, se ha propuesto las redes ópticas elásticas. En estas redes, gracias a la utilización de formatos de modulación eficiente, se permite el establecimiento de supercanales y, con ello, asignar ancho de banda a los usuarios en función de la demanda que soliciten. De esta forma, se puede aprovechar más eficientemente la capacidad de las fibras ópticas. Hay ciertas variaciones en estas redes en cuanto a su configuración, la tasa de transmisión, la modulación o el ancho de los canales [6].

2.1. **Modulación multiportadora**

Para la implementación de redes ópticas elásticas se han propuesto una serie de tecnologías: OFDM (*Orthogonal Frequency Division Multiplexing*), N-WDM (*Nyquist-WDM*) y OAWG (*Optical Arbitrary Waveform Generation*). OFDM es una tecnología prometedora para transmisiones de alta velocidad que tiene características intrínsecas de flexibilidad, es un esquema de modulación multiportadora que transmite un flujo de datos a alta velocidad dividido dentro de cierto número de canales ortogonales, llamados subportadoras. Por otro lado, N-WDM se basa en el uso de pulsos ópticos teniendo un espectro rectangular con un ancho de banda igual a la tasa de símbolos. Estos métodos tienen un rendimiento similar sin embargo OFDM requiere receptores con anchos de banda más grandes y conversores análogo-digital más veloces en respuesta por lo cual N-WDM es más robusto a la hora de implementarse según las restricciones del receptor y la complejidad de los equipos. OAWG es capaz de crear formas de onda de datos con grandes anchos de banda en cualquier formato de modulación. En OFDM se han demostrado transmisiones con un grupo de portadoras especiales espaciadas 12,5 GHz se lograron alcances de 4000 km y con N-WDM se lograron transmisiones de 2800 km y 3200 km con 100 Gbps por canal [7].

2.2. **El problema RSA**

Para el establecimiento de los *lightpaths*, en las redes WRON se necesita la resolución del problema RWA (*Routing & Wavelength Assignment*). En las redes ópticas elásticas el problema equivalente que tiene que ser resuelto es el RSA (*Routing and Spectrum Allocation*) y en caso de hablar también de la selección del formato de modulación trataríamos el problema RMLSA (*Routing, Modulation Level and Spectrum Allocation*). Estos dos problemas son más complejos que RWA puesto que las redes tienen un mayor grado de flexibilidad por lo que el desarrollo de algoritmos eficientes para su resolución es esencial [8, 9].

Cuando se soluciona el problema RSA hay que asegurar la continuidad del espectro a lo largo de los enlaces de una ruta ya definida, esto es, las mismas bandas de frecuencia deben ser usadas a lo largo de ese camino [10]. El problema RSA se puede dividir en dos problemas independientes (el encaminamiento y la asignación del espectro) o resolverlo de forma conjunta. En este TFG nos centramos en resolver el problema de forma independiente. En el encaminamiento hablaríamos de los mismos métodos que en RWA por ejemplo *shortest-path* o *k-shortest paths*. En cuanto a la asignación del espectro se utilizan una serie de métodos como, *first-fit*, *most-used*, *least-used* o *exact-fit*. En RMLSA, además de resolver estos problemas también se determinará el formato de modulación a utilizar para cada *lightpath* [10].

2.3. **Desfragmentación del espectro**

Al establecer las conexiones de manera dinámica según van surgiendo peticiones de *lightpaths*, aparecen de forma inevitable huecos en el espectro que posteriormente serán más difíciles de asignar. Si desfragmentamos el espectro para solucionar este problema,

reorganizaremos los *lightpaths* para hacer hueco a otros que de otra forma serían bloqueados [8, 11]. Existen dos opciones para este planteamiento:

- Reencaminar las rutas de la manera más eficiente para no generar huecos innecesarios entre los anchos de banda ya ocupados (opción A en la *Figura 2*).
- Reasignar los huecos de tal forma que no es necesario cambiar de ruta, sino que los reubicamos dentro de la misma (opción B en la *Figura 2*).

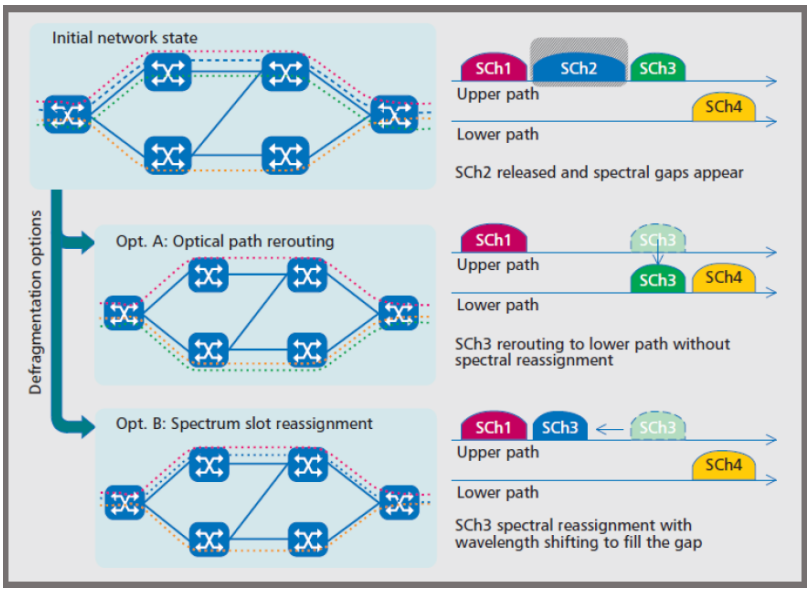

Figura 2. Técnicas de desfragmentación del espectro

Este procesamiento requiere que no se interrumpa el servicio durante la reasignación del espectro.

2.4. Grooming

A la hora de responder a una petición, el nodo de control deberá decidir si encaminar la conexión por la capa eléctrica y/o por la capa óptica, a esto se le denominará *grooming*. En el nivel de la capa eléctrica se determinará como encaminar la nueva conexión a través de una combinación de *lightpaths* nuevos o que estén establecidos en la red, a nivel óptico se decide como establecer estos *lightpaths* cumpliendo con las restricciones de continuidad del espectro [12].

2.5. Fibras multicore

Las redes de comunicaciones ópticas están alcanzando su capacidad límite debido al crecimiento de la demanda de datos en la última década generado por el auge de teléfonos inteligentes, *tablets*, redes sociales y un largo etc. Para solventar este problema se ha propuesto incrementar la capacidad límite de las redes ópticas mediante la sustitución de la fibra óptica "clásica" por la fibra óptica multinúcleo (*MCF, Multi-core fiber*), la cual es capaz de multiplicar la capacidad de estas fibras, pero manteniendo su dimensión [13].

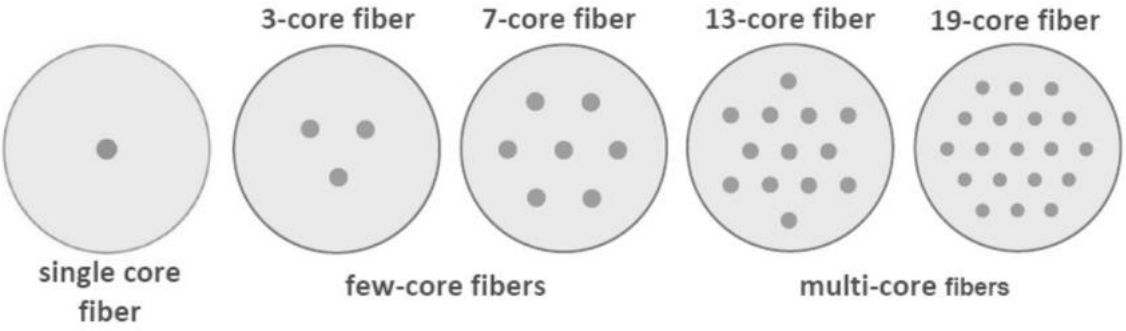

Figura 3. Ejemplo de fibras ópticas multi núcleo

La compañía (Sumitomo Electric Industries) ha desarrollado una fibra óptica de cuatro u ocho núcleos adecuado para interconexiones ópticas, siendo actualmente el cable de fibra óptica de más alta densidad de núcleos [13].

Esta fibra se caracteriza por una atenuación de 0,158 dB/km a una longitud de onda de 1550 nm. La capacidad de esta fibra hace que al ser tan alta dependa de las características de emisores y receptores más que de la propia fibra. Conlleva a mayores que el conjunto de conectores, cables, placas y dispositivos necesarios sean más costosos que en una red de fibra óptica estándar [14].

El uso de estas fibras en las redes óptícas elásticas plantea un nuevo problema, como pasaba con las fibras mononúcleo con el problema RSA, en este caso tenemos el problema RCSA (*Routing, Core & Spectrum Allocation*). En RCSA, para el establecimiento de un *lightpath*, además de determinar la ruta y una fracción del espectro, hay que determinar el *core* por el que estableceremos dicha conexión. Los puntos positivos son el aumento de la capacidad de transmisión, sin embargo, no son todo ventajas hay que tener en cuenta el aumento de la complejidad computacional debido al aumento de una dimensión en el *core* de la fibra y una serie de problemas con la capa física, el *crosstalk* (XT) entre los *cores* en fibras MCF (*Multi-Core Fiber*) tiene un gran impacto ya que la separación entre los *cores* es menor cuanto más aumentamos el número de *cores* a mayores a la limitación del tamaño del revestimiento [16].

Capítulo 3. Simulador de redes ópticas elásticas *multicore*

El simulador de redes WRON desarrollado por el Grupo de Comunicaciones Ópticas de la Universidad de Valladolid utiliza un esquema de control centralizado donde un nodo central gestiona todas las peticiones de *lightpaths* y el estado de la red, en conjunto con una serie de nodos que generan estas peticiones.

Aunque el simulador puede trabajar con cualquier topología, para la realización de las pruebas se han utilizado dos topologías: un anillo metropolitano con una distancia entre nodos de 2 km y la NSFNet de 14 nodos (topología de red de transporte establecida en EEUU y muy utilizada en investigación para la validación de este tipo de métodos). A continuación, se describirán los algoritmos utilizados y las modificaciones realizadas para hacerlos funcionar en una red óptica elástica que utiliza fibras *multicore* entre los distintos nodos de la red para conseguir aumentar el ancho de banda de la misma.

3.1. Modificaciones realizadas al simulador de redes elásticas del GCO

A continuación, se describen las modificaciones realizadas al simulador de redes elásticas del GCO:

1. Implementación automática de una red en anillo indicando el número de nodos y el número de núcleos de la fibra *multicore* utilizada.
2. Adaptación de la topología NSFNet utilizando fibras *multicore*.
3. En el simulador actual, a la hora de resolver el problema del encaminamiento se utiliza el método *k-shortest paths* pero solo era posible utilizar el número de saltos como métrica. Se ha modificado el simulador para que se pudiera utilizar también el retardo de propagación como métrica. Ahora el usuario puede elegir entre estas dos métricas.
4. El simulador de redes elásticas utilizaba como parámetro de rendimiento la probabilidad de bloqueo. Se ha implementado mejoras para que pueda estimar también el retardo de propagación medio y el número medio de *sublightpaths* que se generan.
5. Adaptación de los métodos RSA implementados para ser utilizado en redes que utilizan SDM con fibras *multicore*, es decir, disponer de métodos RCSA.
6. Propuesta de dos metodologías RCSA (basadas en las anteriormente propuestas por el GCO) para la solución del problema RSA en redes que utilizan SDM.

3.2. **Algoritmos RCSA dinámicos**

En una red dinámica, tenemos un conjunto de usuario conectados a distintos nodos de acceso de la red. Estos usuarios irán solicitando el establecimiento de *lightpaths* entre un nodo origen y un nodo destino. Además, en cada petición se indicará el ancho de banda que dichos usuarios reclaman. En el caso de que el *lightpath* pueda ser establecido, dicha conexión se liberará cuando el usuario lo determine.

Cuando una petición de establecimiento de *lightpath* llega a la red, hay que buscar recursos para el establecimiento de la conexión y para ello, hay que utilizar algoritmos qure resuelvan el problema RCSA (*Routing Core and Spectrum Allocation*). Se han propuesto métodos para solucionar el problema RCSA que resuelven de forma independiente el routing y por otro lado, la selección de *core* y espectro. Como método de selección de rutas se ha utilizado *k-shortest paths* implementando las dos métricas anteriormente comentadas: *k-Shortest-Path* (en función de los saltos) y *Least-Delayed-Path* (en función del retardo de propagación).

Para utilizar *k-shortest paths* en una red *multicore* ha sido necesario realizar modificaciones a la implementación que actualmente se encontraba en el simulador de forma que se pueda trabajar con fibras que tengan más de un *core* en la configuración de la red, lo que se traduce en tener más de un enlace disponible con cada salto entre nodos.

Para la selección de core y asignación de espectro, se han propuesto dos métodos basados en el algoritmo de asignación de espectro (sin selección de *cores*) *best-gap* [15]. Este algoritmo, tal y como se demostró en dicha referencia, consigue buenos resultados cuando se compara con una técnica clásica como *first-fit*. Sin embargo, no se puede utilizar directamente en redes con fibras *multicore*. En este TFG se han propuesto dos variaciones a *best-gap* para escenarios *multicore*, es decir para solucionar los problemas CSA:

- *Cores* independientes: el método busca de forma iterativa en cada uno de los *cores* de la ruta utilizando el método *best-gap* para asignar ancho de banda en cada uno de ellos. En el momento en que hay suficientes recursos para establecer la conexión, el proceso de búsqueda se detiene.

- *Cores* expandidos: Se simula que todos los *cores* forman un espectro continuo. En este espectro de continuo se utilizará *best-gap* para la asignación de ancho de banda. Su funcionamiento será similar a utilizar *best-gap* en una red con un único *core* por fibra suponiendo que la capacidad de dicho enlace es *n* veces la capacidad de un *core*, siendo *n* el número de *cores* por fibra.

En todas ellas, hemos supuesto que no hay convertidores de longitud de onda por lo que la asignación de espectro a un *lightpath* debe ser la misma en todos los saltos del camino. También, se ha supuesto que se utilizarán los mismos *cores* a lo largo en todos los saltos del camino siguiendo las conclusiones de [17] donde se demuestra que una arquitectura de ROADM sin soporte de cambio de línea (*line change*) resulta mucho más económica (aunque con una pequeña penalización en el rendimiento de la red) que si utilizamos una arquitectura con dicha funcionalidad.

Para conocer la utilización del espectro en cada *core* de la red, el simulador tiene implementada una TED (*Traffic Engineering Database*) que almacena el estado de los mismos y que responde a las consultas sobre la ocupación de los enlaces necesaria para la resolución del problema RCSA. Cuando se soluciona dicho problema, para la ruta seleccionada se obtiene Matriz de Capacidad Conjunta (MCC) para cada conjunto ruta-*core*. Esta matriz contiene el espectro disponible en dicho par y se forma superponiendo la disponibilidad de espectro en cada par ruta-*core*. Vamos a ilustrar el solapamiento en frecuencia con el siguiente ejemplo.

Suponemos que tenemos tres enlaces que forman la ruta entre dos nodos (enlaces 2, 7 y 9) y vamos a generar la matric MCC para utilizar un *core* determinado (llamémosle *core* c). Las frecuencias ocupadas en el *core* a evaluar en los enlaces del camino son las siguientes:

- Enlace 2 – *Core* c: Frecuencias ocupadas de 100 a 300 GHz
- Enlace 7 – *Core* c: Frecuencias ocupadas de 0 a 100 GHz y de 700 a 800 GHz,
- Enlace 9 – *Core* c: Frecuencias ocupadas de 100 a 500 GHz.

Así pues, las frecuencias no disponibles en el par camino-*core* será: 0 a 500 GHz y de 700 a 800 GHz tal y como observamos, representadas en color azul, en la *Figura 4*. Por tanto, el espectro disponible, según la MCC (y representado en blanco en la *Figura 4*) son las frecuencias entre 500 y 700 GHz, y entre 800 y 1000 GHz.

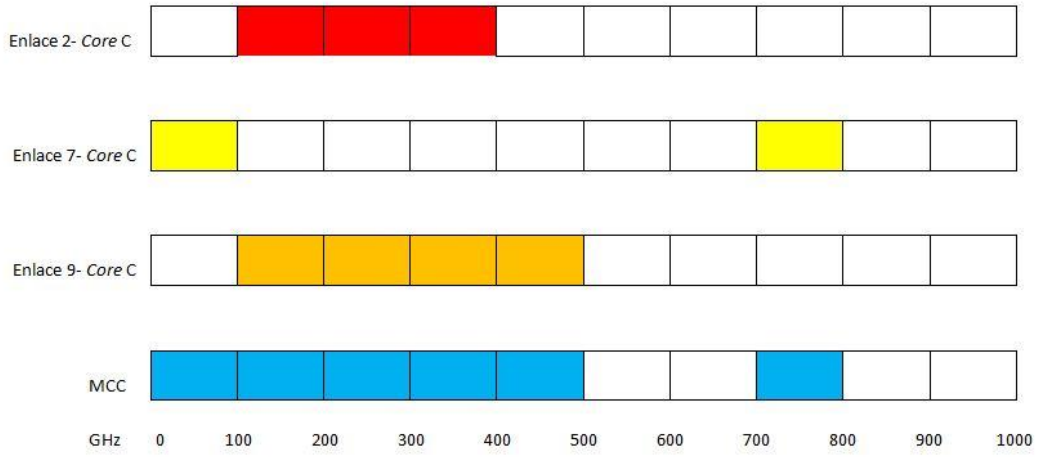

Figura 4. Ejemplo de MCC para ejemplificar el solapamiento en frecuencia

Con las MCC para cada par *ruta-core*, aplicaremos los algoritmos propuestos para la solución del problema RCSA. Estos métodos se describen a continuación.

3.3. **Algoritmos de Selección de Core y Asignación de Espectro**

Antes de comenzar a explicar los algoritmos y el funcionamiento de los mismos los clasificaremos utilizando la misma nomenclatura de [15]:

- De acuerdo a la posibilidad de dividir una demanda en varios *sublightpaths*:
  - *Joint*: El ancho de banda solicitado debe asignarse en un único hueco, es decir no podemos fragmentar el ancho de banda demandado, debe ir unido.
  - *Disjoint*: El ancho de banda designado puede fragmentarse según necesidad y se puede dividir para que encajen en huecos diferentes, incluso pueden estar en rutas diferentes.

Lo ilustraremos con un ejemplo (*Figura 5*), viendo la MCC, con los huecos ocupados en azul y los libres en blanco, se aprecia como se ocuparían los huecos disponibles, queriendo asignar 3 huecos representados en rojo tanto si el algoritmo fuera *joint* o *disjoint*. La explicación es válida para las figuras siguientes (*Figura 6* y *Figura 7*).

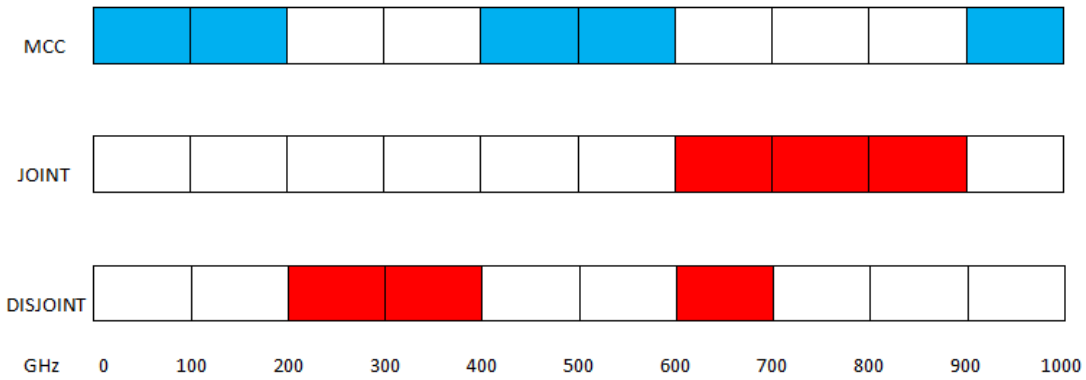

Figura 5. Ejemplo para el uso de Joint y Disjoint

- Uso de rejilla de canales:
  - *Fixed-Grid*: El espectro se fragmenta en slots del mismo tamaño, por lo tanto, las asignaciones se realizan por slots.
  - *Gridless*: El espectro no está fragmentado por lo tanto no hay ningún tipo de restricción a la hora de asignar anchos de banda

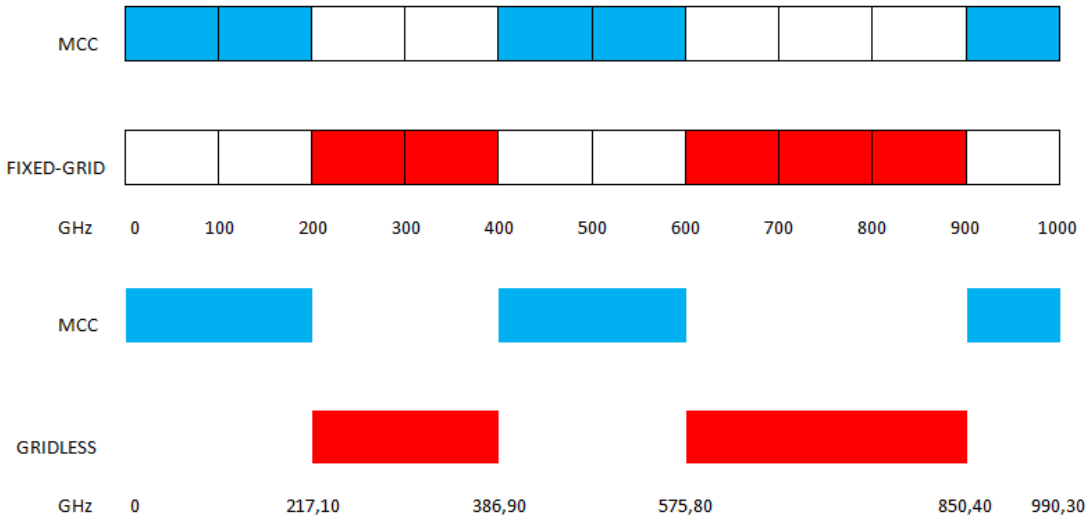

Figura 6. Ejemplo para el uso de Fixed-Grid o Gridless

- Dentro de estas variantes, en [15] se evaluaron dos alternativas para la asignación de espectro,
    - *First*-Fit: Se asignará el primer ancho de banda que se encuentre, empezando por las frecuencias menores.
    - *Best-Gap:* Se asigna el hueco que más adecuado que esté libre, es decir el hueco cuya diferencia entre el tamaño solicitado más el ancho de banda de guarda tenga la menor diferencia posible con el hueco encontrado pero siendo mayor o igual que éste.

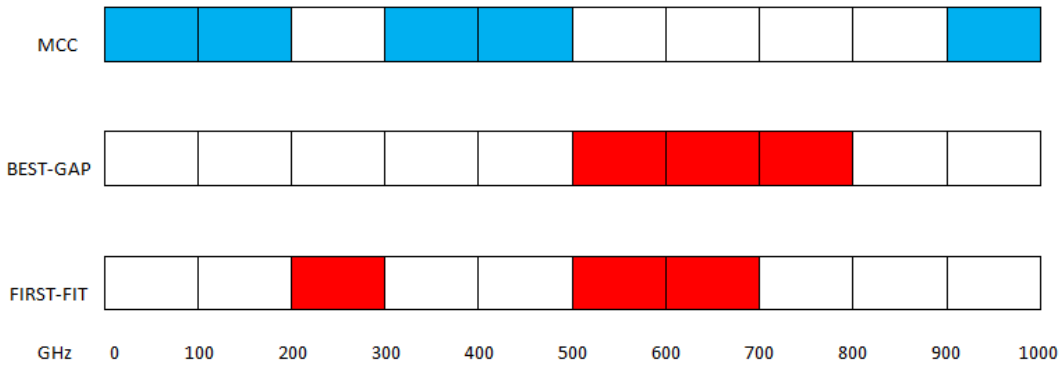

Figura 7. Ejemplo para el uso de Best-Gap y First-Fit con métodos disjoint

Por último, existen alternativas y métodos que además de reservar los recursos para establecer el *lightpath* solicitado utilizan técnicas de protección para conseguir la supervivencia de la red en caso de fallo.

Como se demostró en [15], para conseguir aprovechar la ventaja de las redes flexibles ó elásticas, es necesario utilizar algoritmos eficientes como *best-gap*. Por eso, en este TFG se ha utilizado este método para la evaluación del simulador de redes ópticas elásticas *multicore*. Además, se utilizará la rejilla de canales puesto que es la solución más extendida en la literatura de redes elásticas (aunque de acuerdo a [15], los métodos *gridless* utilizando *best-gap* eran los que conseguían mejores resultados). En cuanto a la posibilidad de dividir la demanda en varios *sublightpaths*, se utilizarán ambas opciones para evaluar cuál funciona mejor en redes *multicore*. De acuerdo a esto, en este TFG, nos centraremos en los siguientes algoritmos, todos ellos sin protección:

- *Joint* Spectrum Fixed-Grid (Best-Gap)
- *Disjoint* Spectrum Fixed-Grid (Best-Gap)

Estos algoritmos tendrán un añadido además de todo lo mencionado anteriormente, la configuración de los *cores* de la red, como ya se ha mencionado, el objetivo de este proyecto es hacer funcionar estos algoritmos en redes *multicore* es decir solucionar el problema RCSA. De esta forma, se han implementado las siguientes alternativas

(además del uso de redes con un único *core* que ya estaba implementada y utilizaremos para comparar):

- *Cores* independientes con Joint y Disjoint Spectrum Fixed-Grid y siempre utilizando la estrategia Best-Gap.
- *Core* expandido con Joint y Disjoint Spectrum Fixed-Grid y siempre utilizando la estrategia Best-Gap.

3.3.1. Joint Spectrum Fixed-Grid (Best-Gap)

En primer lugar, señalar que este algoritmo es *joint*, por lo tanto si hablamos en términos de uso del espectro va a ser mejor que los algoritmos *disjoint* ya que asignaremos menos bandas de guarda (porciones de espectro no utilizable entre conexiones adyacentes) al no fragmentar el espectro. Sin embargo, necesitará encontrar más espectro libre y consecutivo que los métodos *disjoint*

Al tener que utilizar la metodología *best-gap,* debemos conocer los huecos disponibles para saber cuál sería el más adecuado para el ancho de banda demandado. Para esto en cada ruta tendremos una matriz que almacena los huecos disponibles con el más grande en primer lugar y el más pequeño en el último, Matriz de Huecos Libres (MHL).

Ahora analizaremos lo que sucede en este punto en función de si utilizamos una configuración de *cores* independientes o *cores* expandidos.

En caso de tener una configuración de *cores* independientes se actuaría de la siguiente manera:

Para cada camino dentro de los *k*-caminos encontrados ordenados de menor a mayor en función del número de saltos o del retardo de propagación (parámetro seleccionable por el usuario):

1) Por cada *core* ordenados en función de un índice:
    i) Buscar una sección de espectro compacta y disponible en todas las fibras con un ancho de banda mayor del requerido en la petición más la banda de guarda. Se utilizará el método *best-gap* para ello.
    ii) Si se encuentra, establecer el *lightpath* y se terminan las iteraciones para buscar en el resto de *cores*.

Si no se ha establecido el *lightpath*, bloquea la conexión.

Si tuviéramos una configuración con *cores* expandidos, en cada enlace se supone que se tiene un espectro disponible colocando de forma consecutiva los espectros de cada uno de los *cores*. Después se ejecuta la búsqueda de forma similar a la búsqueda en un único *core*:

Para cada camino dentro de los *k*-caminos encontrados ordenados de menor a mayor en función del número de saltos o del retarde de propagación (parámetro seleccionable por el usuario):

1) Buscar una sección de espectro compacta y disponible en todas las fibras con un ancho de banda mayor del requerido en la petición más la banda de guarda. Se utilizará el método *best-gap* para ello.
    i) Si se encuentra, establecer el *lightpath* y se terminan las iteracciones para buscar en el resto de *k*-caminos

Si no se ha establecido el *lightpath*, bloquea la conexión.

En la figura que tenemos a continuación vemos como los slots libres están indicados en verde y los ocupados en naranja, intentaremos asignar 2 slots (flechas azules) junto con el ancho de banda de guarda (flecha amarilla) correspondiente, el espectro supondremos que está compuesto por 13 slots:

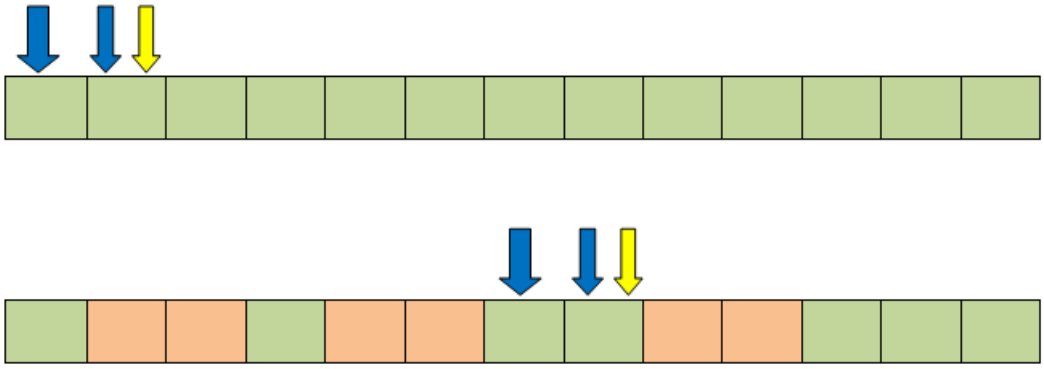

Figura 8. Espectro en el caso de JSFG-BG

3.3.2. Disjoint Spectrum Fixed-Grid (Best-Gap)

El funcionamiento de este algoritmo es similar al caso *joint*, con la diferencia de que al ser un algoritmo *disjoint* podremos fragmentar el ancho de banda demandado, es decir todo el ancho de banda demandado no tiene porque asignarse por una sola ruta, por lo que concatenaremos rutas hasta que suplamos el ancho de banda demandado.

Analizaremos lo que ocurre si tenemos una configuración de *cores* independientes o de *cores* expandidos.

En caso de tener una configuración de *cores* independientes se actuaría de la siguiente manera:

Para cada camino dentro de los *k*-caminos encontrados ordenados de menor a mayor en función del número de saltos o del retardo de propagación (parámetro seleccionable por el usuario):

1) Por cada *core* ordenados en función de un índice:
    i) Buscar una sección de espectro disponible en todas las fibras con un ancho de banda mayor del requerido en la petición más la banda de guarda, aquí debemos distinguir dos casos:
        (1) Se encuentra un hueco que según el criterio *best-gap* sea suficiente para suplir el ancho de banda a establecer. Se establece el *lightpath* (utilizando las reservas de espectro que se han efectuado para la

conexión incluyendo la que se acaba de encontrar). Se acaba el algoritmo.

(2) El hueco encontrado es menor al ancho de banda a establecer, por lo tanto se reserva el ancho de banda que nos sea posible en el hueco más grande disponible en ese *core*, y reduciría el ancho de banda a establecer en una cantidad al ancho de banda de hueco encontrado (teniendo en cuenta el *gap*). El proceso continúa para tratar de encontrar ancho de banda suficiente para la petición.

Si no se ha establecido el *lightpath*, se bloquea la conexión y se liberan las reservas.

Si tuviéramos una configuración con *cores* expandidos, en cada enlace se supone que se tiene un espectro disponible colocando de forma consecutiva los espectros de cada uno de los *cores*. Después se ejecuta la búsqueda de forma similar a la búsqueda en un único *core*:

Para cada camino dentro de los *k*-caminos encontrados ordenados de menor a mayor en función del número de saltos o del retarde de propagación (parámetro seleccionable por el usuario):

1) Buscar una sección de espectro y disponible en todas las fibras con un ancho de banda mayor del requerido en la petición más la banda de guarda aquí debemos distinguir dos casos:
   a) Se encuentra un hueco que según el criterio *best-gap* sea suficiente para suplir el ancho de banda a establecer. Se establece el *lightpath* (utilizando las reservas de espectro que se han efectuado para la conexión incluyendo la que se acaba de encontrar). Se acaba el algoritmo.
   b) El hueco encontrado es menor al ancho de banda a establecer, por lo tanto se reserva el ancho de banda que nos sea posible en el hueco más grande disponible en ese *core*, y reduciría el ancho de banda a establecer en una cantidad al ancho de banda de hueco encontrado (teniendo en cuenta el *gap*). El proceso continúa para tratar de encontrar ancho de banda suficiente para la petición.

Si no se ha establecido el *lightpath*, se bloquea la conexión y se liberan las reservas.

Mostraremos el funcionamiento del mismo con el siguiente ejemplo, tenemos un espectro de 13 slots (slots libres en verde y ocupados en naranja), queremos asignar 4 slots (flechas azules) y la banda de guarda (flechas amarillas), debido a que no es posible asignarlos de manera consecutiva acabaremos asignando 5 slots al tener que asignar los slots en los distintos huecos que encontremos:

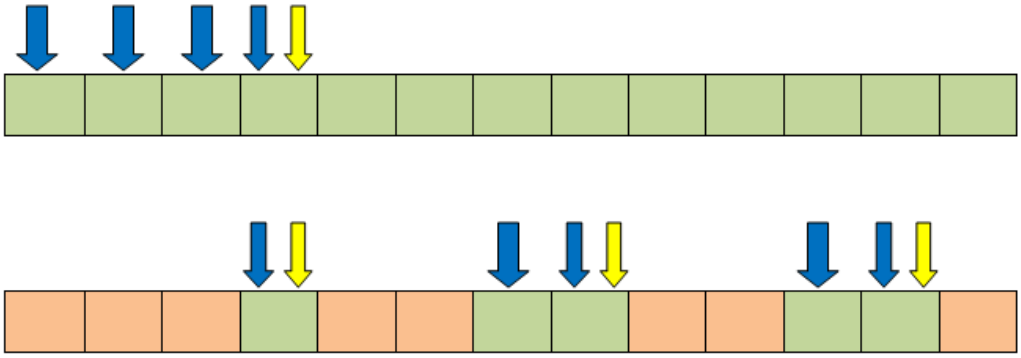

Figura 9. Espectro de ejemplo en DSFG-BG

Capítulo 4. Evaluación de los métodos propuestos utilizando el simulador desarrollado

A continuación se muestran los resultados obtenidos utilizando el simulador implementado en la plataforma OMNeT++. Con estos resultados tratamos de averiguar cuál de los algoritmos es más eficiente mediante el análisis de los siguientes parámetros:

- Probabilidad de bloqueo,
- Retardo de propagación medio de los *lightpaths*.
- Número de *sublightpath* establecidos (para los métodos *disjoint*)
- Tiempo medio de computación.

Para ellos, se han realizado pruebas con distintas configuraciones de parámetros:

- Redes utilizadas:
  - Anillo metropolitano de 14 nodos, con una distancia entre nodos de 2 km.
  - NSFNet de 14 nodos, cuya configuración viene descrita en [18].
- Numero de transmisores y receptores por nodo: 1000
- Ancho de banda disponible por *core* 3750 GHz
- Número de *cores*: 1 y 2.
- Carga: Variaremos la carga desde 0,1 hasta 0,9, esta carga la variaremos en intervalos de 0,1
- Ancho de banda de guarda: 10 GHz. Para todas las configuraciones de tamaños de *slots*, se utilizará el mismo ancho de banda de guarda para tener una comparativa equitativa.
- Las peticiones piden anchos de banda aleatorios según una distribución uniforme entre 1 GHz y 300 GHz.

En cuanto a la configuración de los métodos implementados se ha utilizado:

- Numero de caminos (*k*): 2 en el anillo y NFSNet. En la NSFNet este valor podría ser mayor pero se ha decidido mantener el mismo para comparar ambas redes y la influencia del grado lógico de los nodos.
- Orden de las rutas: Las rutas entre los distintos orígenes y destinos se ordenan al inicializar la red, en este caso ordenamos las rutas por el criterio *k-Shortest-Paths,* es decir se utilizan en primer lugar las que tengan un menor número de saltos entre origen y destino.
- Tamaño del slot: En nuestro caso al utilizar únicamente algoritmos *Fixed-Grid* variamos el tamaño de slot entre los siguientes valores: 12,5 GHz, 25 GHz, 50 GHz, 100 GHz y 200 GHz.

A continuación, se mostrarán los resultados para la red en anillo metropolitano con distintas configuraciones y posteriormente haremos una comparativa con la red NSFNet. Todos los resultados se muestran en media con su intervalo de confianza del 95%.

4.1. **Red en Anillo - Joint Spectrum Fixed-Grid (Best-Gap)**

En primer lugar, consideraremos la red en anillo. Analizaremos los datos obtenidos para el algoritmo *Joint Spectrum Fixed-Grid* y observaremos cuál de los tamaños de slot es más eficiente para este caso utilizando las rutas ordenadas por *shortest-path*, el número de caminos estará establecido en 2, barreremos la carga.

Dentro de este algoritmo tenemos que hacer tres distinciones para el análisis de los datos, primero revisaremos este algoritmo utilizando un *core*, posteriormente con dos *cores* expandidos, después con dos *cores* independientes y por último realizaremos una comparativa entre los mismos.

4.1.1. JSFG-BG 1 Core

Mostramos la probabilidad de bloqueo (*Figura 10*), el tiempo medio de computación (*Figura 11*) y el retardo de propagación medio por *lightpath* (*Figura 12*) en las siguientes figuras:

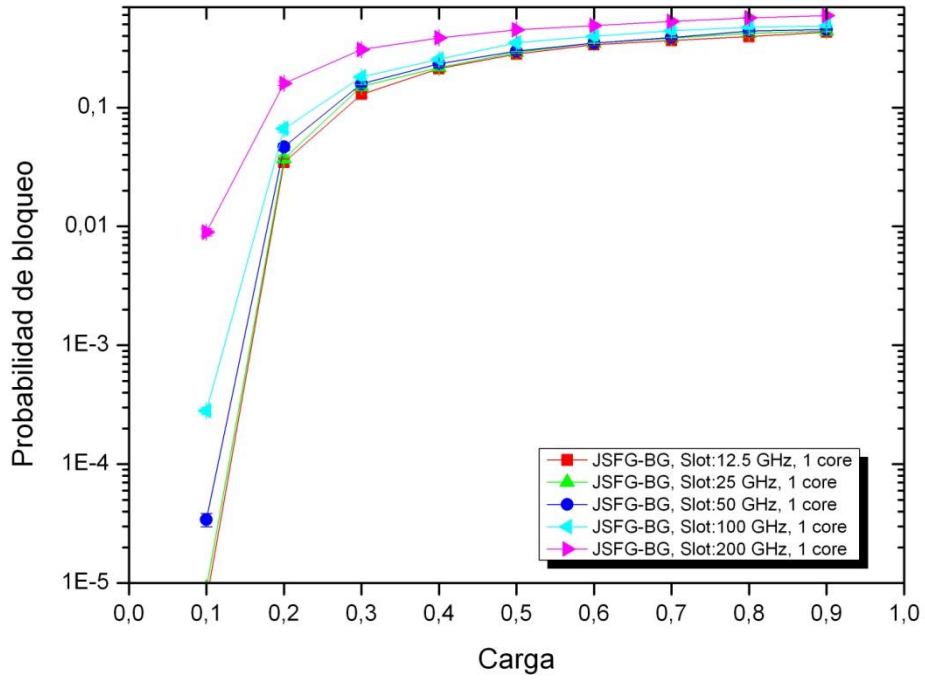

Figura 10.Red en Anillo, Probabilidad de bloqueo del algoritmo JSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

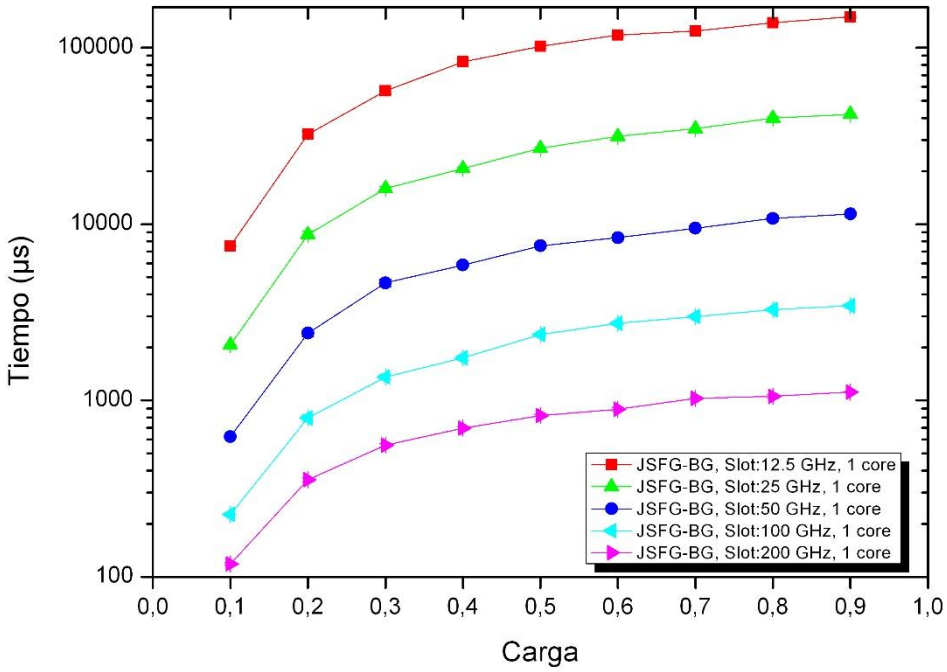

Figura 11. Red en Anillo, Tiempo de computación del algoritmo JSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

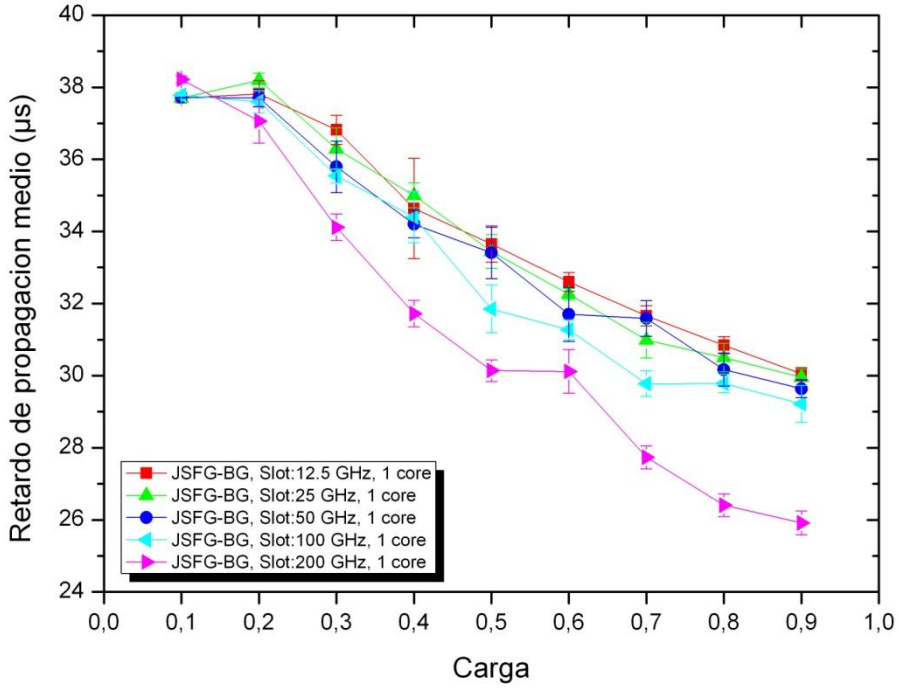

Figura 12. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo JSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Apreciamos en la *Figura 10* que las probabilidades de bloqueo decrecen con la carga de la red. Además, podemos observar que las probabilidades de bloqueo más bajas las obtenemos para los tamaños de slot más pequeños, 12,5, 25 y 50 GHz, aunque como contrapartida en la *Figura 11* podemos ver que el tiempo de computación para los algoritmos de 12,5, 25 y 50 GHz son bastante mayores que con un tamaño de slot de 200 o 100 GHz (dos órdenes de magnitud superiores en algunas cargas), esto se debe a que cuanto menor sea el tamaño de slot más iteraciones necesitaremos para asignar el ancho de banda solicitado ya que el espectro esta fragmentado en más partes y se requiera más tiempo de búsqueda. A pesar de esta contrapartida buscamos reducir la probabilidad de bloqueo lo máximo posible.

En cuanto a la *Figura 12* vemos que el retardo de propagación medio es mayor para los tamaños de slot 12,5, 25 y 50 GHz, esto indica que los *lightpaths* establecidos son más largos en cuanto a los saltos entre nodos que abarcan, esto se debe a que la gestión del espectro es mejor cuanto menor es el tamaño de los slots utilizados, ya que al tener el espectro fragmentado en trozos más pequeños podemos asignar el ancho de banda de manera más eficiente y por lo tanto, se permite el establecimiento de *lightpaths* más largos.

En las comparativas de las distintas alternativas utilizaremos los tamaños de slot de 12,5, 25 y 50 GHz ya que son los que nos ofrecerán unos mejores resultados en cuanto a probabilidad de bloqueo, aunque esto ofrezca peores resultados en otros parámetros.

4.1.2. JSFG-BG 2 Cores Expandidos

Mostraremos los resultados de la probabilidad de bloqueo (*Figura 13*), tiempo de computación (*Figura 14*) y retardo de propagación medio (*Figura 15*).

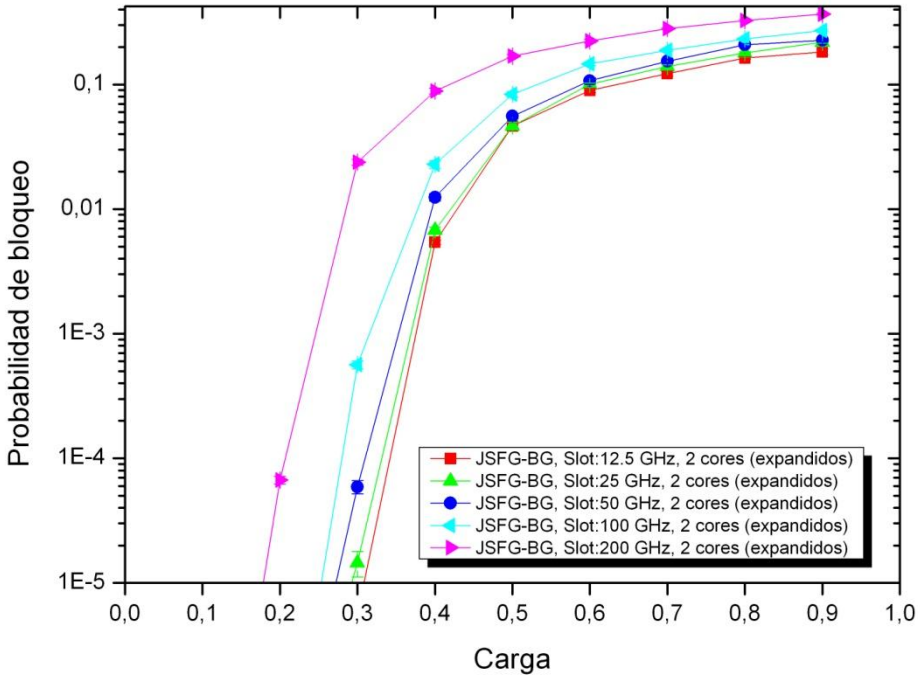

Figura 13. Red en Anillo, Probabilidad de bloqueo del algoritmo JSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

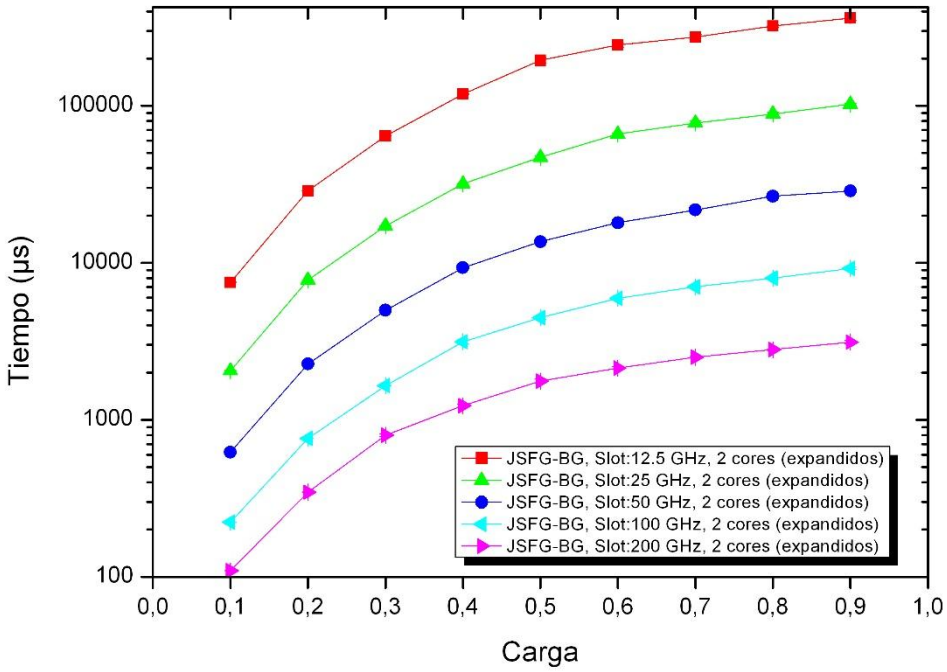

Figura 14. Red en Anillo, Tiempo de computación del algoritmo JSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

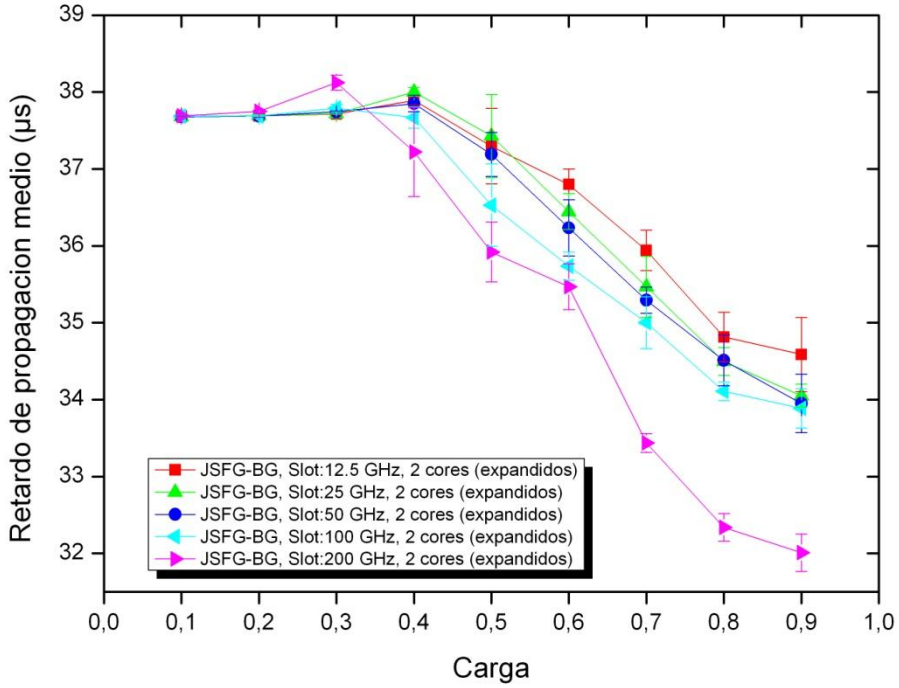

Figura 15. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo JSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Las conclusiones que podemos extraer de estas gráficas son las mismas que en la configuración de 1 *core*, en la probabilidad de bloqueo (*Figura 13*) se obtienen los mejores resultados cuanto menor es el tamaño de slot ya que hacemos un uso más eficiente de la capacidad del espectro.

El tiempo de computación (*Figura 14*) muestra que cuanto menor es el tamaño de slot mayor es el tiempo de computación ya que tenemos el espectro más fragmentado y será necesario realizar más iteracciones para asignar el ancho de banda necesario para establecer el *lightpath*.

En la *Figura 15*, podemos observar cómo en cargas bajas de 0,1 y 0,2 la duración de los *lightpaths* para todos los tamaños de slot es la misma. Esto se debe a que al tener el doble de capacidad en la fibra (7500 GHz en lugar de 3750 GHz) en cargas bajas nos posibilita el establecer estos *lightpaths* de la misma longitud en saltos entre nodos.

En la comparativa entre los distintos métodos utilizaremos los resultados de 12,5, 25 y 50 GHz, ya que como en el caso anterior son los que ofrecen los mejores resultados respecto a la probabilidad de bloqueo.

4.1.3. JSFG-BG 2 Cores Independientes

A continuación, se muestran los resultados en término de la probabilidad de bloqueo (*Figura 16*), tiempo de computación (*Figura 17*) y retardo de propagación medio (*Figura 18*) como en los casos anteriores.

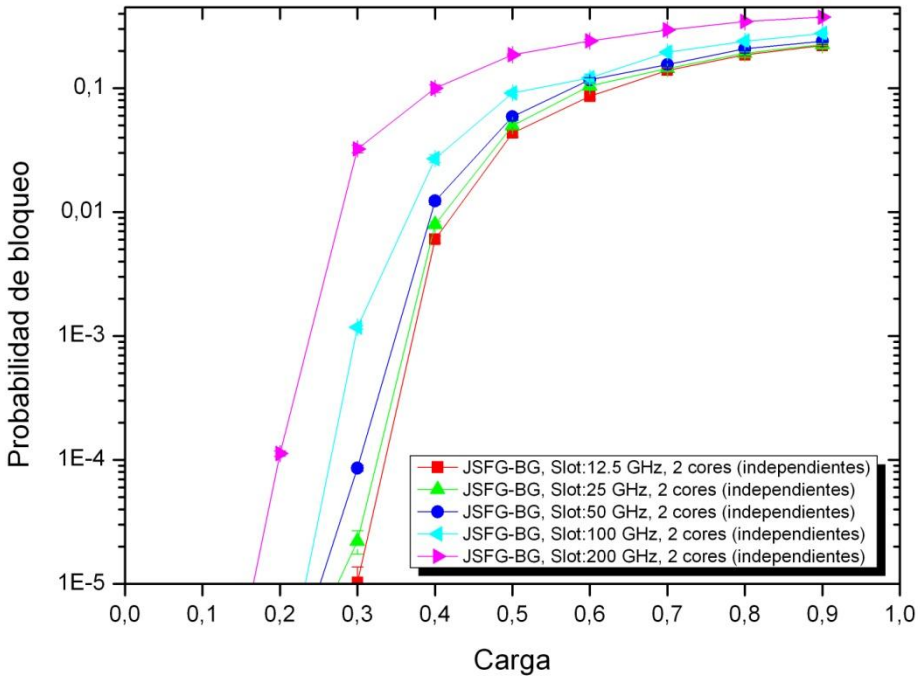

Figura 16. Red en Anillo, Probabilidad de bloqueo del algoritmo JSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

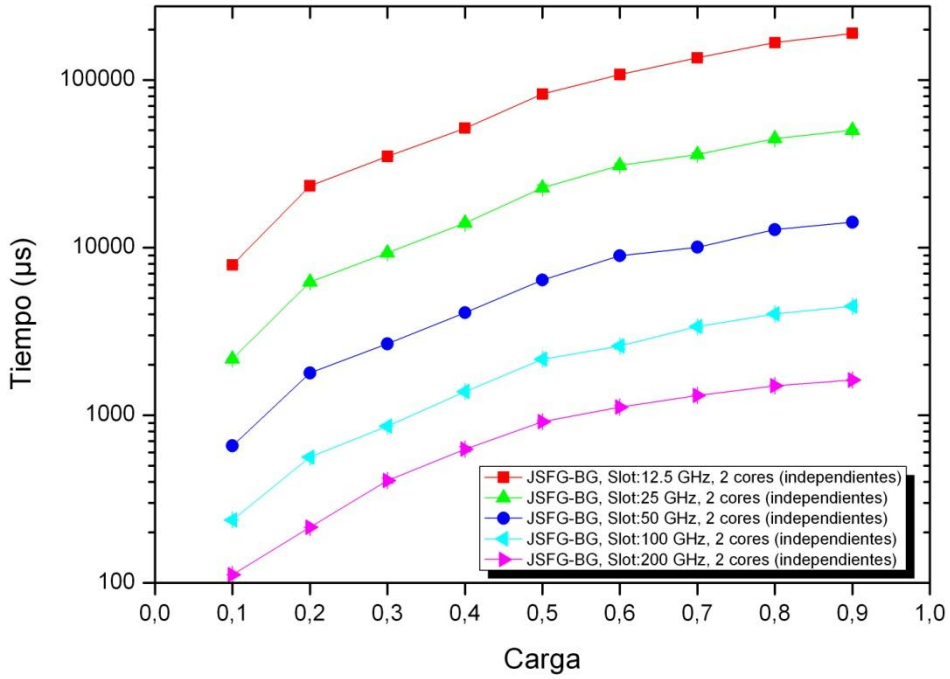

Figura 17. Red en Anillo, Tiempo de computación del algoritmo JSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

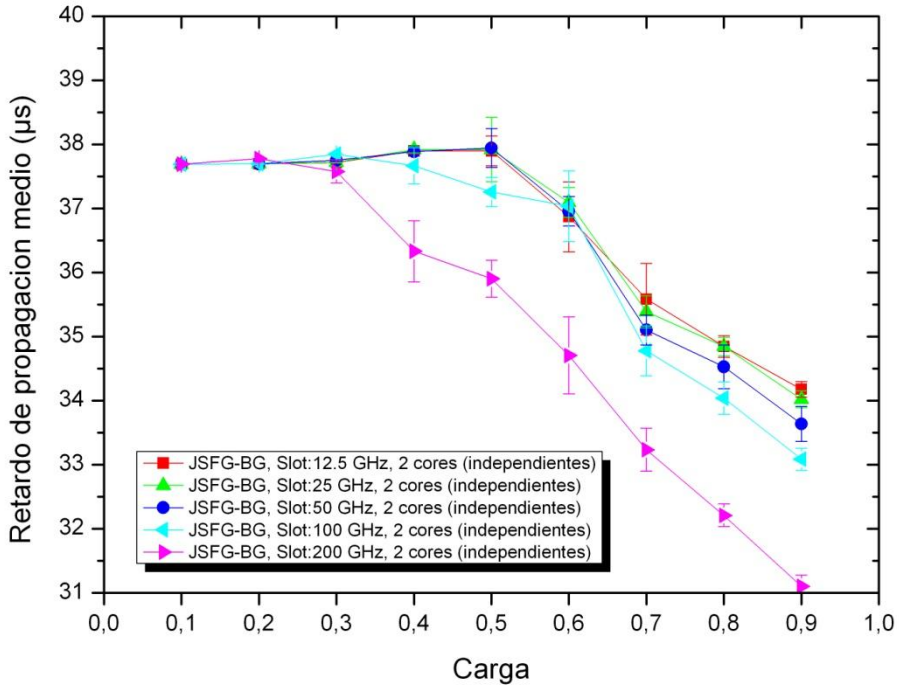

Figura 18. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo JSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Los tamaños de slot que ofrecen la probabilidad de bloqueo (*Figura 16*) menor son 12,5, 25 y 50 GHz ya que se hace un uso más eficiente del espectro y asignar la capacidad demandada de forma más eficiente, debido a tener el espectro divido en slots más pequeños.

Como podemos observar en el tiempo de computación (*Figura 17*) cuanto menor es el tamaño de slot el tiempo de computación es mayor ya que tenemos que realizar un mayor número de iteraciones para poder asignar la capacidad demandada al tener el espectro fragmentado en slots más pequeños.

En cuanto al retardo de propagación medio por *lightpath* (*Figura 18*) observamos que los *lightpaths* establecidos cuanto menor es el tamaño de slot más largos (en número de saltos) son en todas las cargas, esto se debe a una gestión más eficiente del espectro al tenerlo fragmentado en slots más pequeños y una asignación más eficiente de la capacidad demandada.

4.1.4. Comparativa Algoritmos JSFG-BG

Realizaremos una comparativa entre los algoritmos JSFG-BG con las distintas configuraciones y veremos cual ofrece un mejor rendimiento en cuanto a la probabilidad de bloqueo (*Figura 19*). También analizaremos las diferencias entre el tiempo medio de computación (*Figura 20*) y el retardo de propagación medio (*Figura 21*).

Recordamos que utilizamos el criterio *shortest-path*, con un número de caminos (*k*) de 2 en todos los casos. Comenzaremos analizando cuál de las dos configuraciones entre la de 1 *core* y 2 *cores* independientes es más eficiente.

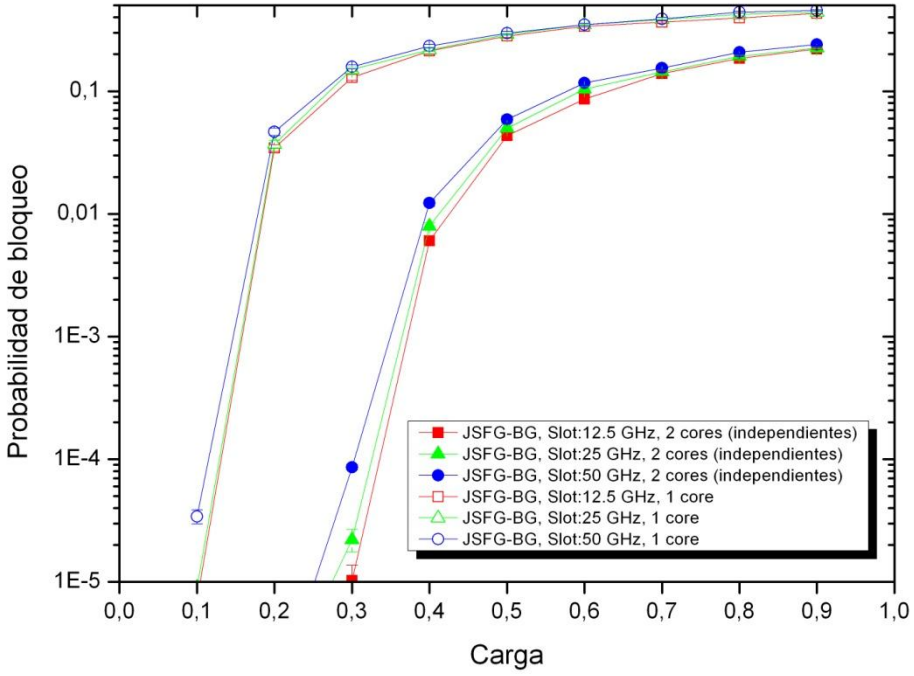

Figura 19. Red en Anillo, Probabilidad de bloqueo del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

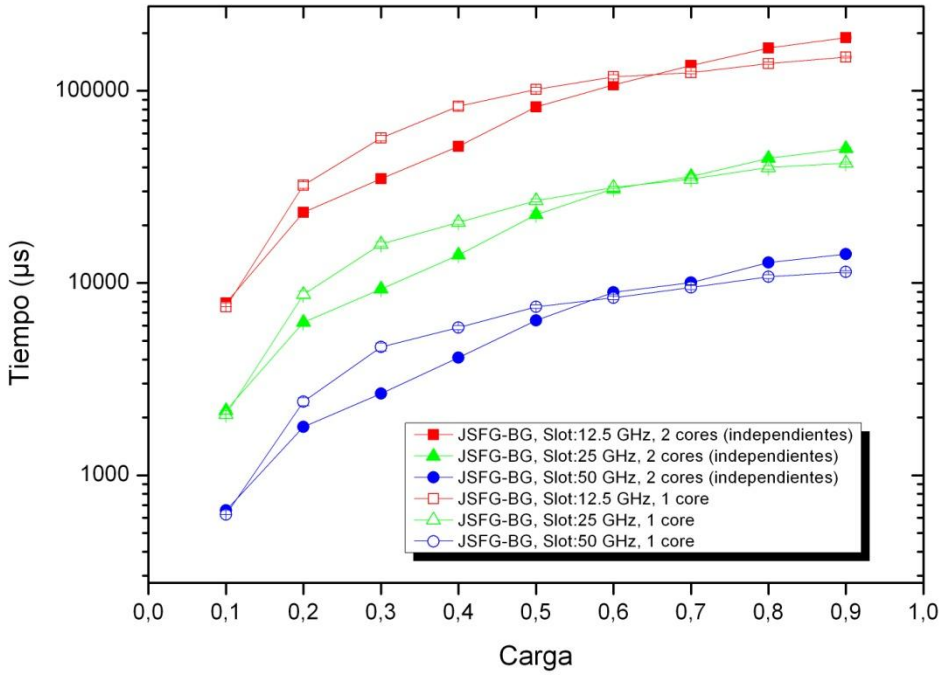

Figura 20. Red en Anillo, Tiempo de computación del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

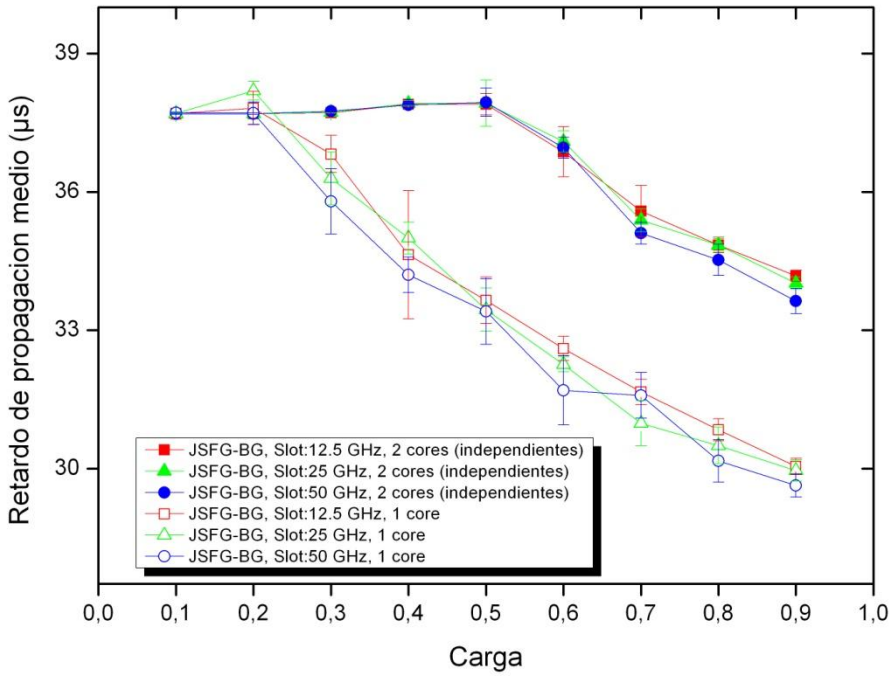

Figura 21. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Como se puede observar claramente la configuración de 2 *cores* independientes ofrece unos resultados mejores en cuanto a la probabilidad de bloqueo (*Figura 19*), ya que como podemos ver para trabajar con una probabilidad de bloqueo de 0,1 aproximadamente en el caso de 1 *core* deberíamos trabajar con una carga de la red de 0,3, sin embargo con la configuración de 2 *cores* independientes podríamos trabajar con cargas de 0,7.

En cuanto al tiempo de computación (*Figura 20*) podemos observar que para cargas altas la configuración de 2 *cores* independientes obtiene peores resultados ya que tiene que realizar más iteraciones al tener más posibilidades para establecer los *lightpaths* cumpliendo con el ancho de banda demandado, sin embargo obtenemos tiempos de computación menores cuando la carga disminuye de 0.6, esto se debe a que si tenemos poca carga en el caso de la configuración de 1 *core* si no podemos establecer el *lightpath* por la ruta más corta se pasará comprobar la ruta más larga lo que requerirá un mayor tiempo de computación. Por el contrario, si utilizamos la configuración de 2 *cores* independientes en el caso de no poder establecer el *lightpath* por la ruta más corta en el primer *core* pasaríamos a comprobar esta misma ruta por el segundo *core*, por lo tanto el número de comprobaciones que haremos será menor que en el caso de tener que revisar la ruta larga como en el caso de 1 *core*.

El retardo de propagación medio (*Figura 21*) nos indica que la configuración de 2 *cores* independientes nos permite establecer *lightpaths* más largos debido a poder utilizar más de un *core* en la misma ruta para la asignación de la capacidad demandada.

Podemos deducir, como era de esperar, que la configuración de 2 *cores* independientes ofrece mejores resultados en la probabilidad de bloqueo y dependiendo de la carga de la red mejores tiempos de computación.

Viendo que esta es la mejor configuración de las dos, vamos a proceder a compararla con la configuración de 2 *cores* expandidos y veamos cuál ofrece mejores resultados.

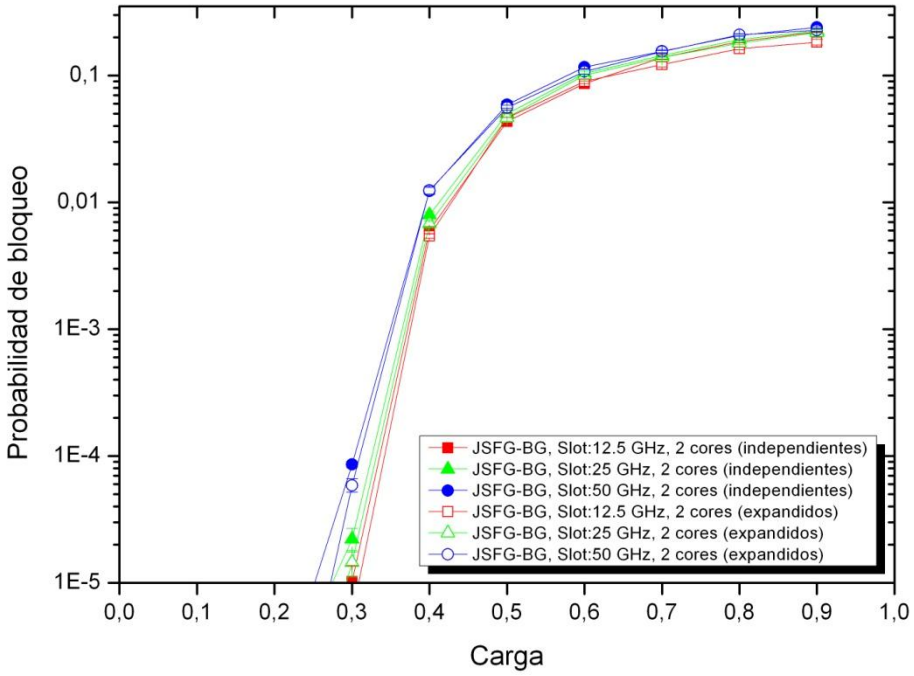

Figura 22. Red en Anillo, Probabilidad de bloqueo del algoritmo JSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

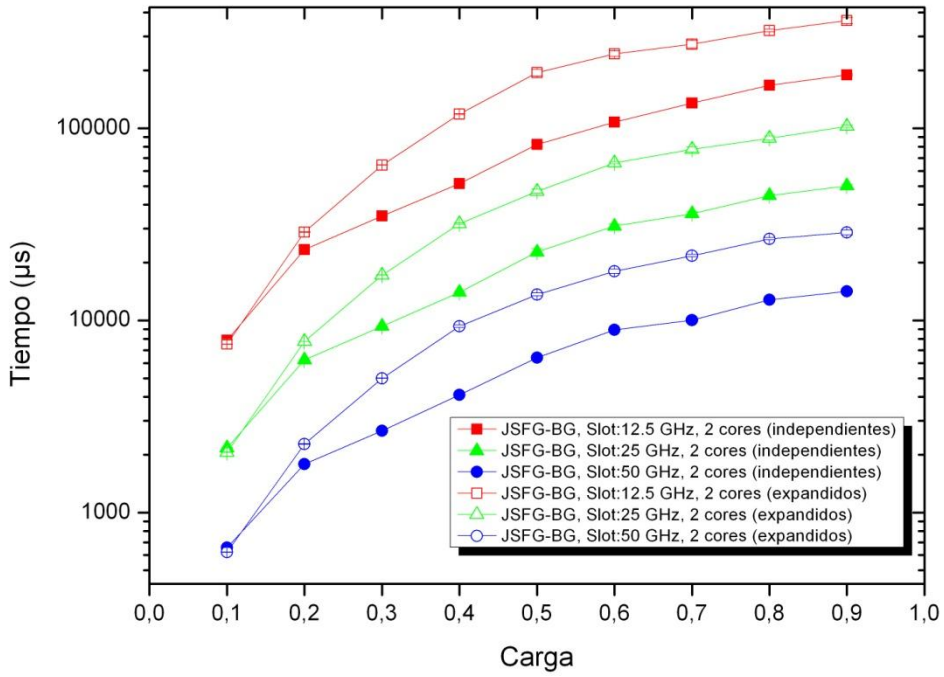

Figura 23. Red en Anillo, Tiempo de computación del algoritmo JSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Como podemos observar en términos de probabilidad de bloqueo ambas configuraciones ofrecen resultados muy similares aunque cuando las cargas bajan de 0,3 la configuración de 2 *cores* expandidos ofrece unos resultados algo mejores. Que los resultados sean tan parejos se explica de una manera muy simple, con estas dos configuraciones al tener un algoritmo JSFG-BG buscaremos el hueco que mejor se adapten a la capacidad demandada independientemente de la capacidad de los *cores* y del número de los mismos que tengamos, por lo tanto aprovecharemos el espectro de la misma forma.

En cuanto al tiempo de computación podemos observar que la configuración de 2 cores independientes los valores obtenidos respecto a la configuración de 2 *cores* expandidos son menores, la explicación sería la misma que comparando la configuración de 1 *core* y 2 *cores* expandidos que debido a tener más rutas cortas disponibles para establecer los *lightpaths* entre los distintos nodos tendremos que hacer menos comprobaciones de los estados de los enlaces y por lo tanto el tiempo de computación será menor.

Por lo tanto, viendo los resultados obtenidos podríamos deducir que la configuración más eficiente es la configuración de 2 *cores* expandidos para el algoritmo JSFG-BG si solo nos fijamos en la probabilidad de bloqueo.

4.2. **Red en Anillo - Disjoint Spectrum Fixed-Grid (Best-Gap)**

Analizaremos los resultados para este algoritmo DSFG-BG, con ordenación de rutas según *k-shortest paths* con un número de caminos máximo (*k*) de 2 y seleccionando el hueco más adecuado según el criterio *best-gap*. Obtendremos los parámetros de la probabilidad de bloqueo, tiempo de computación, retardo medio de propagación y un nuevo parámetro que llamaremos número medio de *sublightpaths* por conexión. Con este nuevo parámetro mediremos la cantidad de veces que se fragmenta la capacidad demandada por las distintas rutas disponibles para establecer el *lightpath* solicitado ya que estamos utilizando un algoritmo *disjoint* y podemos fragmentar la capacidad demandada en las distinitos *sublightpaths* utilizando *traffic grooming*.

Este algoritmo lo utilizaremos con las configuraciones de 1 *core*, 2 *cores* expandidos y 2 *cores* independientes. Posteriormente haremos una comparativa entre estas y veremos cuál es la más eficiente.

4.2.1. DSFG-BG 1 Core

Analizaremos los parámetros mencionados anteriormente de la probabilidad de bloqueo (*Figura 24*), tiempo de computación (*Figura 25*), retardo medio de propagación (*Figura 26*) y el número medio de *sublightpaths* por conexión (*Figura 27*).

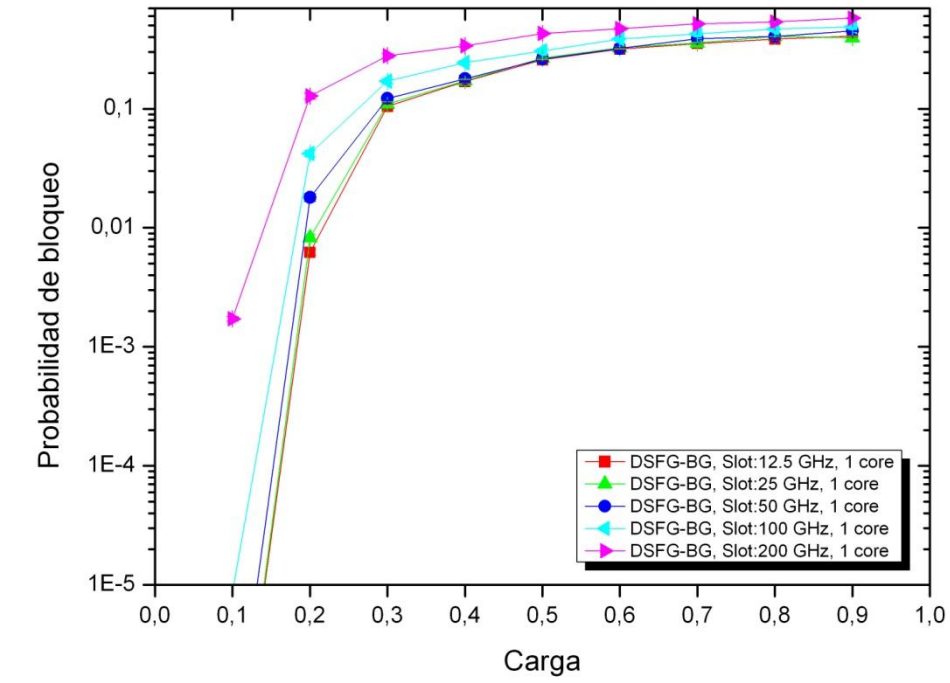

Figura 24. Red en Anillo, Probabilidad de bloqueo del algoritmo DSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

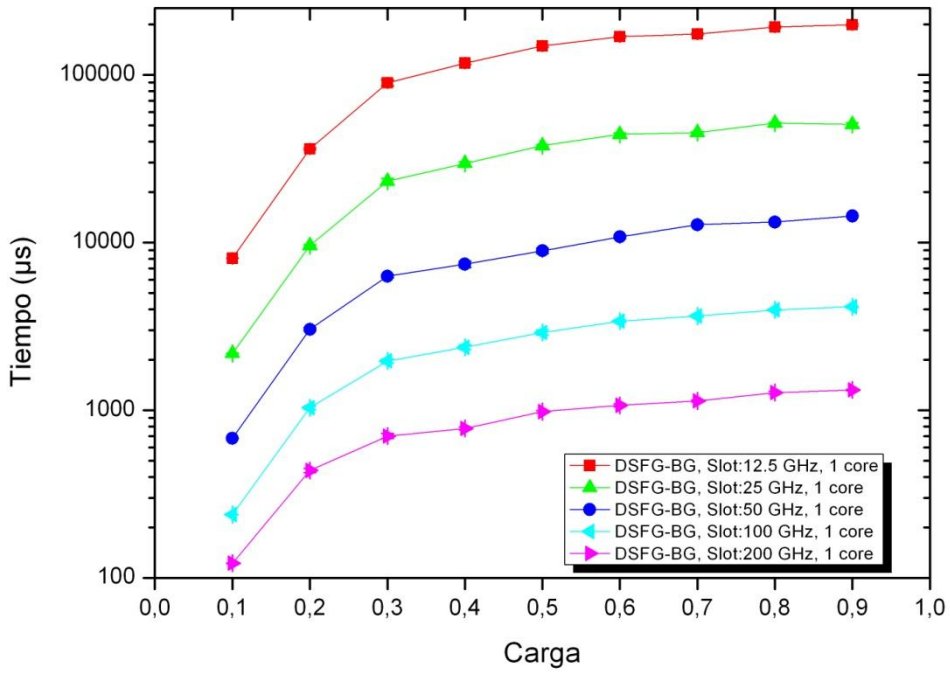

Figura 25. Red en Anillo, Tiempo de computación del algoritmo DSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

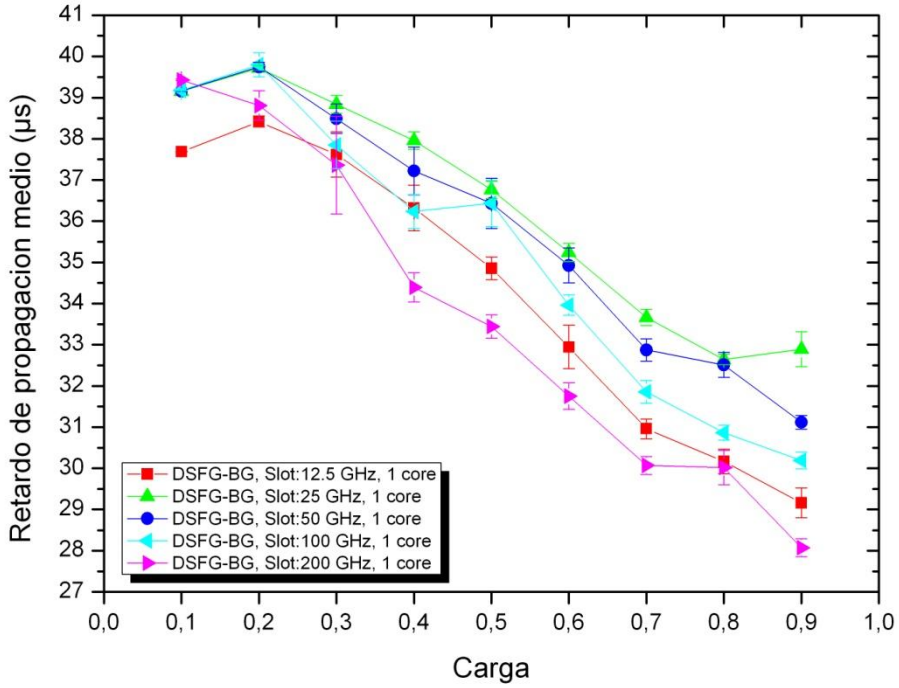

Figura 26. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

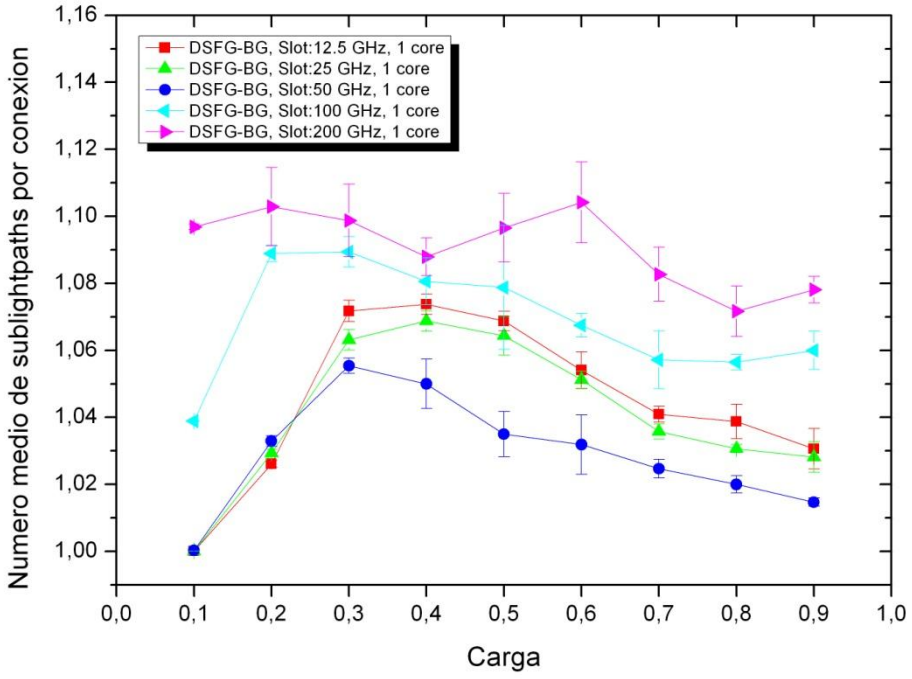

Figura 27. Red en Anillo, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (1 core) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En cuanto a la probabilidad de bloqueo en la *Figura 24*, observamos que los tamaños de slot más pequeños de 12,5, 25 y 50 GHz ofrecen los mejores resultados como en el algoritmo JSFG-BG.

Las conclusiones de la gráfica del tiempo de computación (*Figura 25*) son equivalentes a los obtenidos en el algoritmo JSFG-BG, cuanto menor es el tamaño de slots mayor es el tiempo de computación ya que tenemos el espectro más fragmentado y se necesita un mayor número de iteraciones para asignar la capacidad demandada.

El retardo de propagación medio (*Figura 26*), nos indica que cuanto mayor es la carga menor es la longitud (en número de saltos) de los *lightpaths* establecidos. Esto se debe a la mayor saturación del espectro al tener una mayor demanda.

En cuanto al número medio de *sublightpaths* por conexión en la *Figura 27* podemos observar que en los tamaños de slot más pequeños (12,5, 25 y 50 GHz) fragmentamos menos que con los tamaños más grandes (100 y 200 GHz). Esto se debe a que con los tamaños más pequeños de slot podemos hacer un uso más eficiente de la capacidad de los enlaces y no será necesario dividir la capacidad solicitada en diferentes rutas para establecer el *lightpath*.

4.2.2. DSFG-BG 2 Cores Expandidos

Comprobaremos los mismos parámetros que en los anteriores casos para ver que tamaños de slot son los más eficientes para esta configuración.

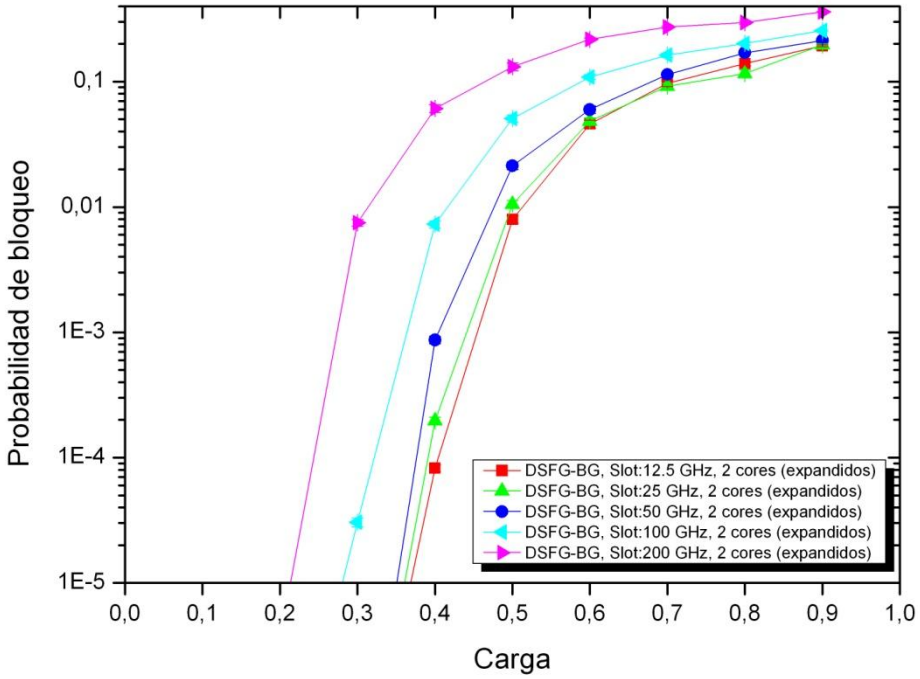

Figura 28. Red en Anillo, Probabilidad de bloqueo del algoritmo DSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

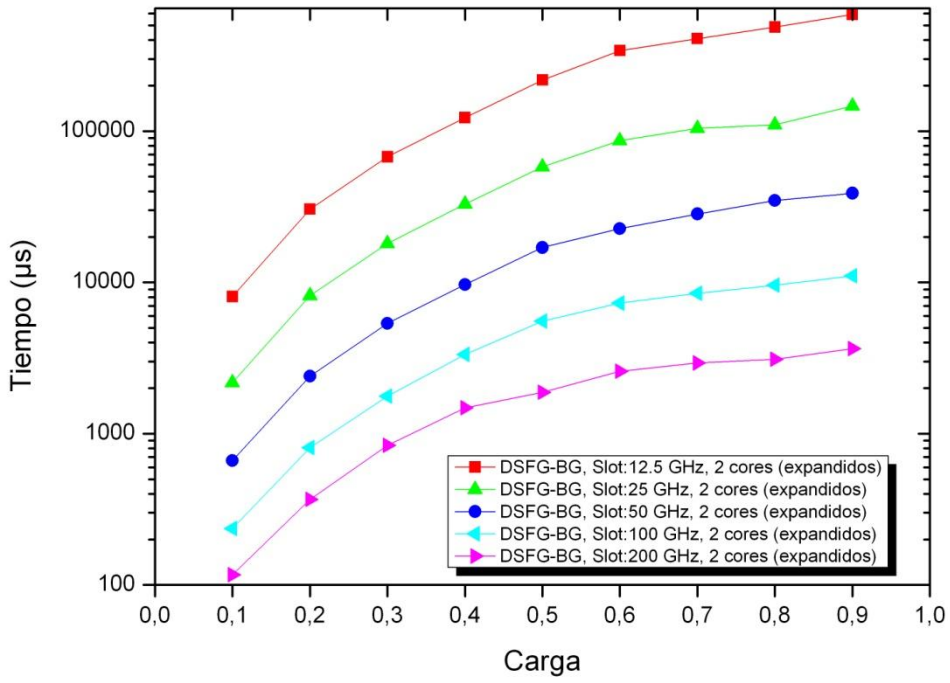

Figura 29. Red en Anillo, Tiempo de computación del algoritmo DSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

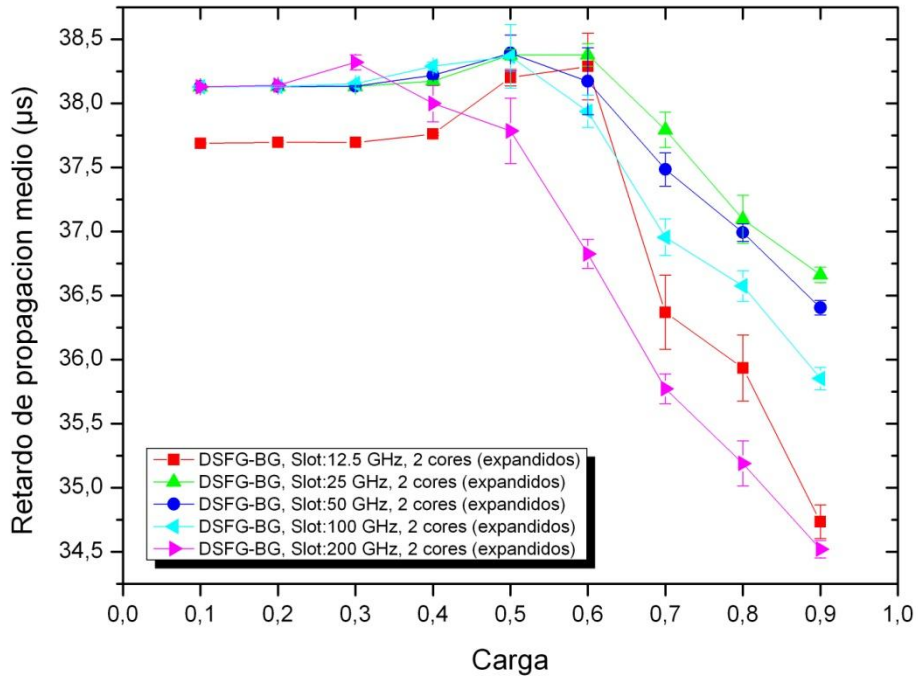

Figura 30. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

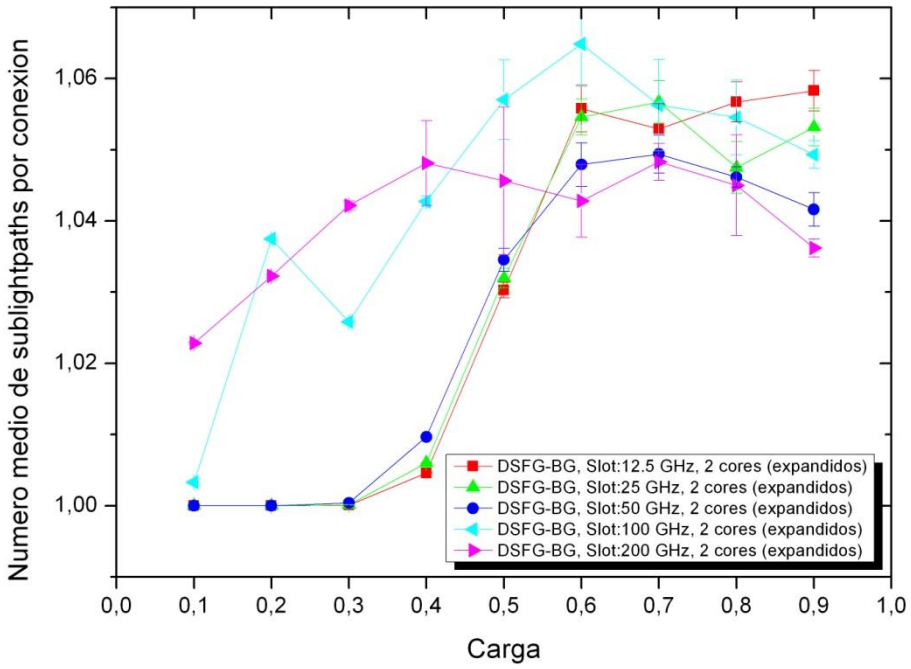

Figura 31. Red en Anillo, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En cuanto a la probabilidad de bloqueo en la *Figura 28*, los tamaños de slot más pequeños (12,5, 25 y 50 GHz) son los que ofrecen unos mejores resultados, ya que se hace un uso más eficiente de la capacidad del espectro.

Respecto al tiempo de computación (*Figura 29*) observamos que cuanto menor sea el tamaño de slot mayor es el tiempo de computación ya que al tener el espectro fragmentado en slots pequeños se han de realizar un mayor número de iteraciones para poder asignar la capacidad demandada.

El retardo de propagación medio (*Figura 30*) nos indica lo mismo que en casos anteriores, con los tamaños de slot de 25, 50 y 100 GHz son los que establecen *lightpaths* más largos (en número de saltos) debido a una gestión más eficiente del espectro al utilizar slots más pequeños.

Viendo el número medio de *sublightpaths* por conexión (*Figura 31*) se observa que se con esta configuración de 2 *cores* expandidos se divide menos la capacidad demandada para poder establecer el *lightpath* que en el caso de 1 *core*, ya que como tenemos más capacidad en cada *core* establecer más *lighpaths* sin tener que fragmentar la capacidad demandada.

4.2.3. DSFG-BG 2 Cores Independientes

Para esta configuración veremos los mismos parámetros que en los casos anteriores, probabilidad de bloqueo (*Figura 32*), tiempo de computación (*Figura 33*), retardo de propagación medio (*Figura 34*) y número medio de *sublightpaths* por conexión (*Figura 35*).

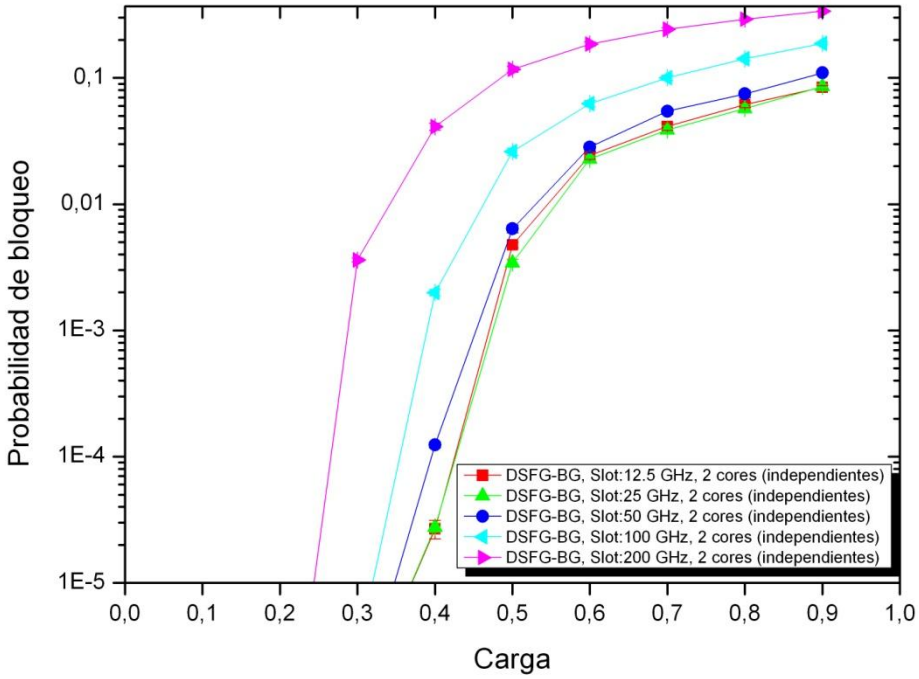

Figura 32. Red en Anillo, Probabilidad de bloqueo del algoritmo DSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

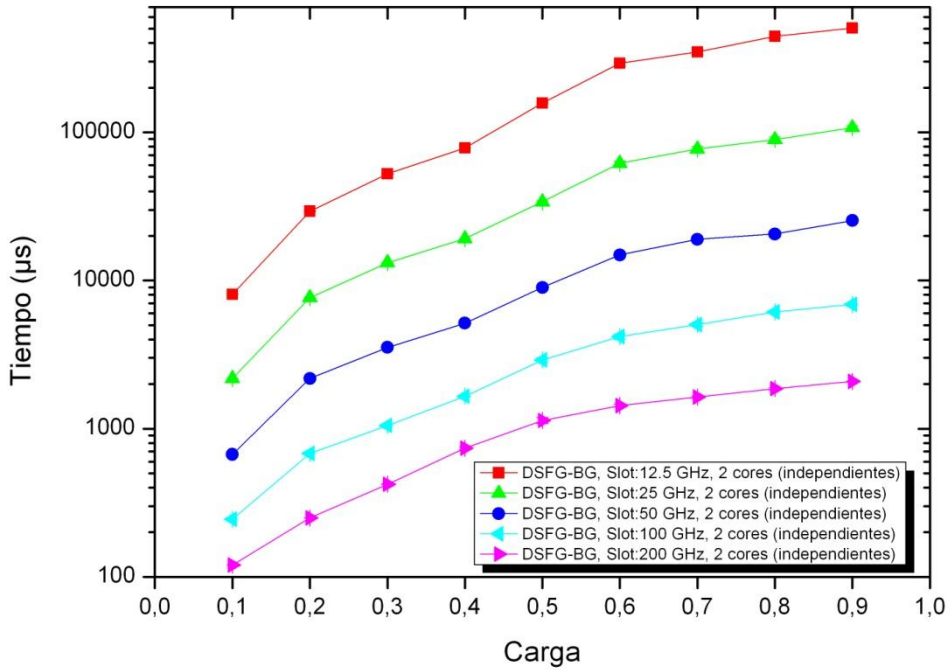

Figura 33. Red en Anillo, Tiempo de computación del algoritmo DSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

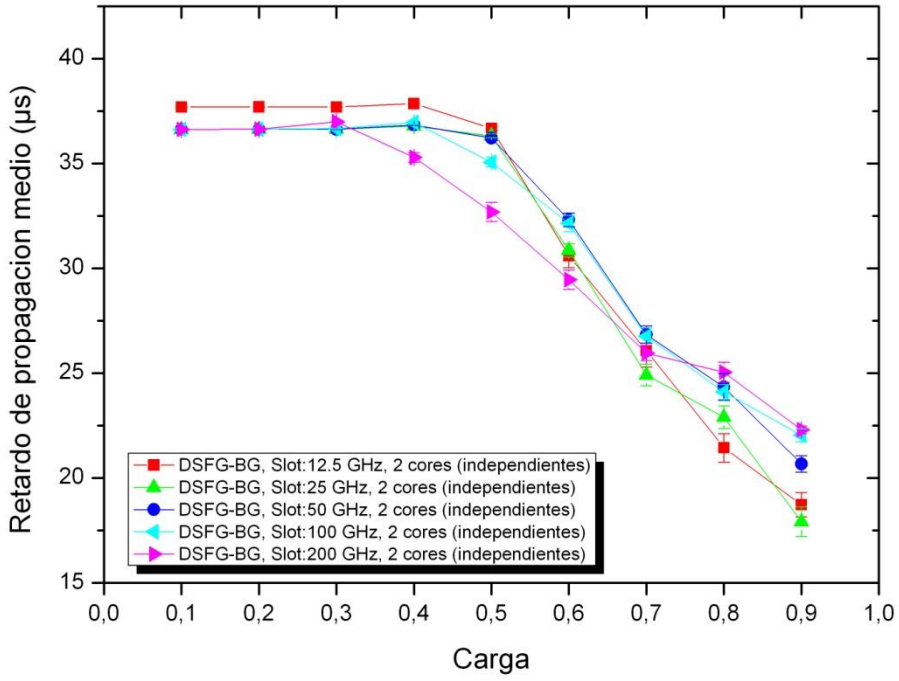

Figura 34. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

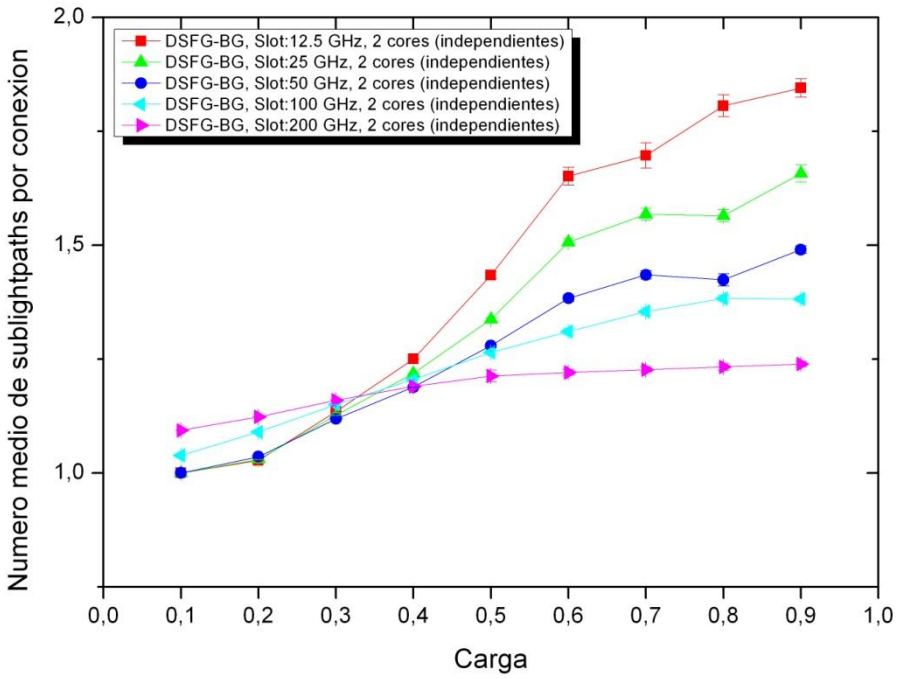

Figura 35. Red en Anillo, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En la *Figura 32* observamos que la probabilidad de bloqueo sigue siendo mejor para los tamaños de slot más pequeños ya que hacemos un uso más eficiente de la capacidad de los *cores* al tener el espectro dividido en slots más pequeños.

El tiempo de computación en la *Figura 33* presenta los resultados esperados, ya que como en los casos anteriores cuanto menor es el tamaño de slot mayor es el tiempo de procesamiento ya que se tendrán que realizar más iteracciones para poder asignar la capacidad demandada tal y como se dijo en la explicación del algoritmo.

En cuanto al retardo de propagación medio (*Figura 34*) se observa que en cargas entre 0,1 y 0,4 todos los *lightpaths* establecidos tienen la misma longitud aproximadamente, en el resto de carga vemos como este valor baja lo que nos indica que al tener cargas más altas habrá *lightpaths* de una longitud mayor que en cargas más bajas se aceptarían pero en cargas altas se rechazan.

La *Figura 35* nos indica que al tener 2 *cores* cuanta mayor es la carga más hay que dividir la capacidad demandada por diferentes rutas para poder establecer los *lightpaths* y por lo tanto este valor aumenta, lo que también se relaciona con el mayor tiempo de procesamiento cuanta mayor sea la carga.

Con estas conclusiones vemos que como ya hemos mencionado los tamaños de slot de 12,5, 25 y 50 GHz son los más eficientes y los que usaremos en la posterior comparativa.

4.2.4. Comparativa Algoritmos DSFG-BG

Comenzaremos la comparativa como en el caso del algoritmo JSFG-BG, comparando la configuración de 1 *core* con 2 *cores* expandidos, utilizando los parámetros de la probabilidad de bloqueo (*Figura 36*), tiempo de computación (*Figura 37*), retardo de propagación medio (*Figura 38*) y el número medio de *sublightpaths* por conexión (*Figura 39*). Utilizaremos el criterio *best-gap*, con ordenación de rutas según *k-shortest paths* y numero de rutas (k) de 2.

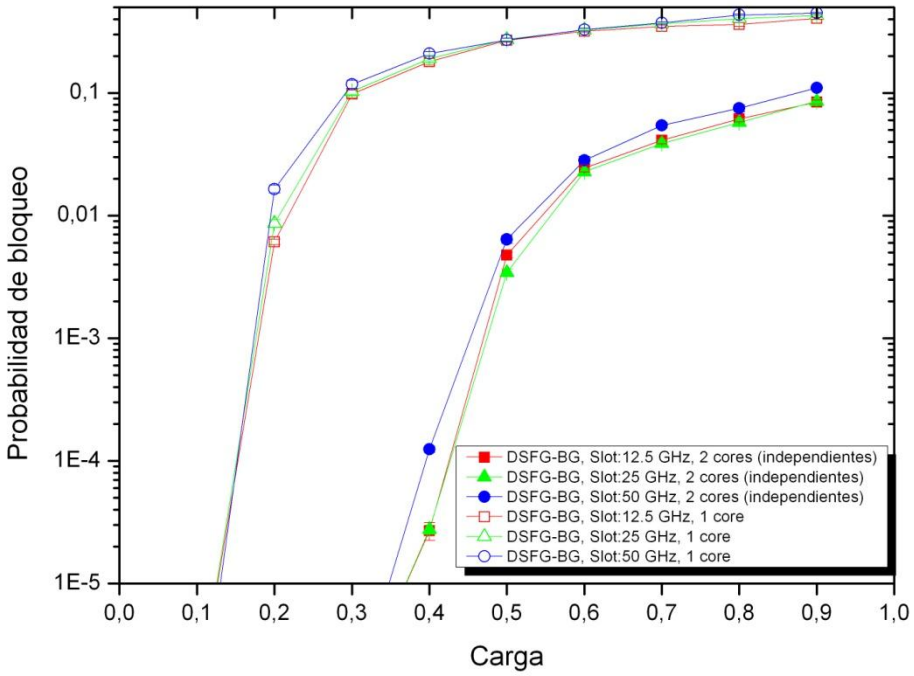

Figura 36. Red en Anillo, Probabilidad de bloqueo del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

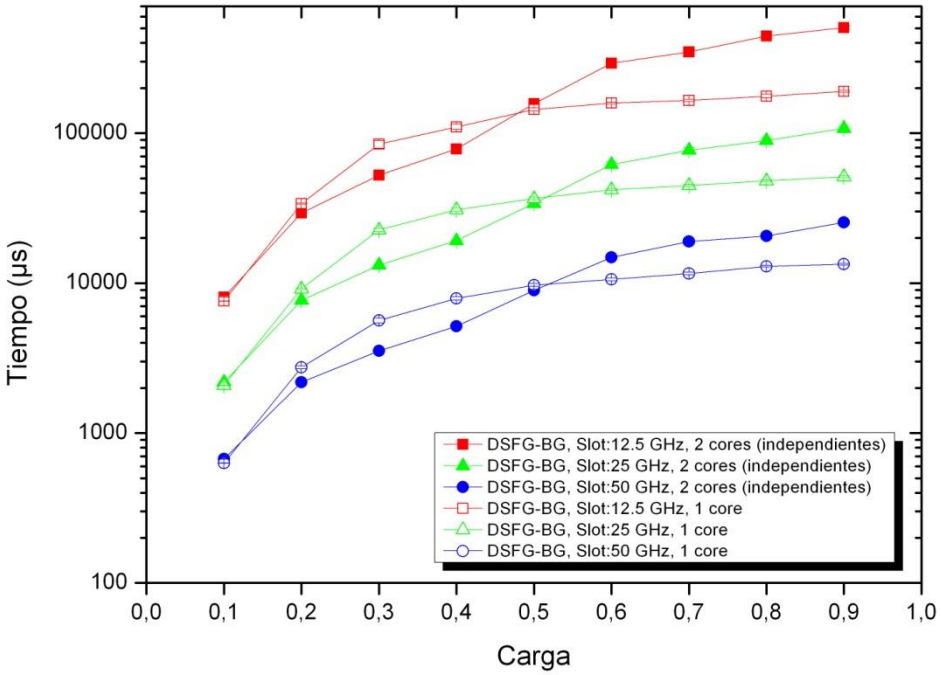

Figura 37. Red en Anillo, Tiempo de computación del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

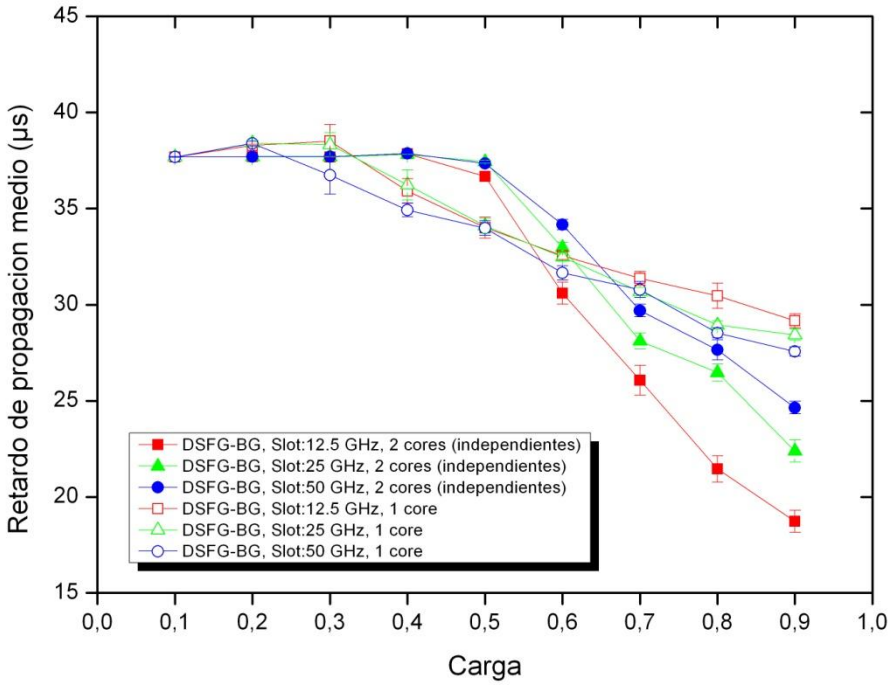

Figura 38. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

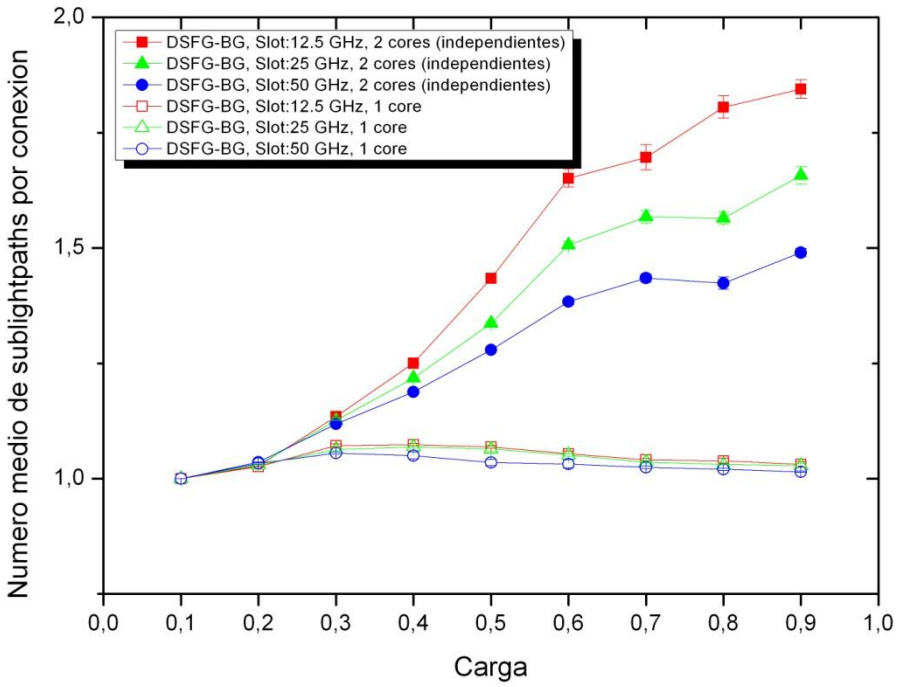

Figura 39. Red en Anillo, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En la *Figura 36* la probabilidad de bloqueo muestra los resultados esperados como sucedía con el algoritmo JSFG-BG, la configuración de 2 *cores* independientes ofrece unos mejores resultados en cualquier valor de carga que tengamos. Se observa que para valores de la probabilidad de bloqueo de 0,01 en la configuración de 1 *core* trabajamos con carga de valor 0,2 y en la configuración de 2 *cores* independientes trabajamos con carga de valor 0,5.

El tiempo de computación (*Figura 37*) ofrece las mismas conclusiones que en la comparativa del algoritmo JSFG-BG, con cargas altas las posibilidades para establecer los *lightpaths* son mayores en el caso de la configuración de 2 *cores* independientes por lo tanto el tiempo de computación será mayor, pero en cargas inferiores a 0,5 la configuración de 2 *cores* independientes ofrece tiempos de computación menores ya que en esta configuración se utilizaran rutas más cortas que en el caso de 1 *core* ya que tenemos más capacidad disponible para asignar en las rutas más cortas.

La *Figura 38* del retardo de propagación medio nos indica que la configuración de 2 *cores* independientes permite establecer *lighpaths* de la misma longitud hasta cargas de 0,5 y este valor va descendiendo según aumentamos la carga y en el caso de 1 *core* solo se mantendría la misma longitud hasta cargas de 0,3 y volverán a descender estos valores según aumentemos la carga.

El valor del número medio de *sublightpaths* por conexión (*Figura 39*) muestra los resultados esperados, en la configuración de 2 *cores* independientes dividiremos más la capacidad demandada en caso de ser necesario ya que las posibles rutas entre nodos son

mayores que en el caso de 1 *core* y ofrecería más opciones en caso de ser necesaria esta división.

Al igual que en el caso del algoritmo JSFG-BG, la configuración más eficiente es la de 2 *cores* independientes y a continuación lo compararemos con la configuración de 2 *cores* expandidos.

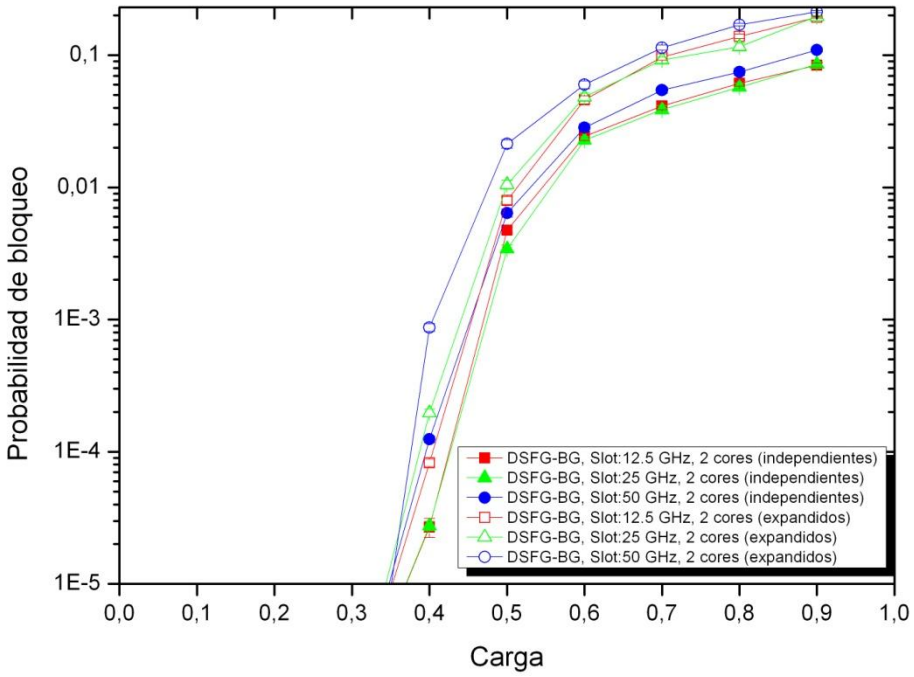

Figura 40. Red en Anillo, Probabilidad de bloqueo del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

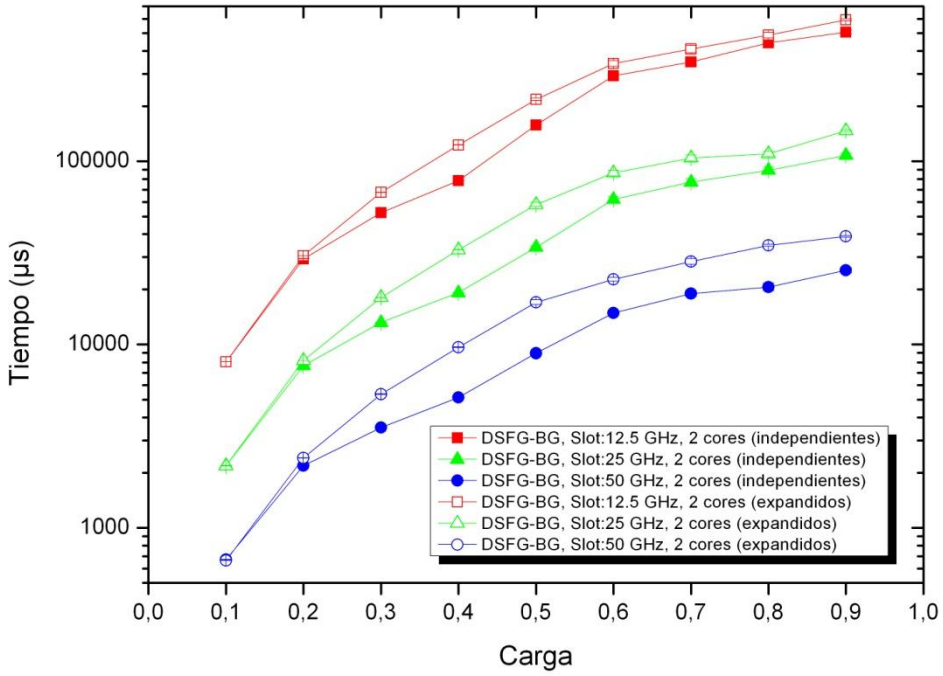

Figura 41. Red en Anillo, Tiempo de computación del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

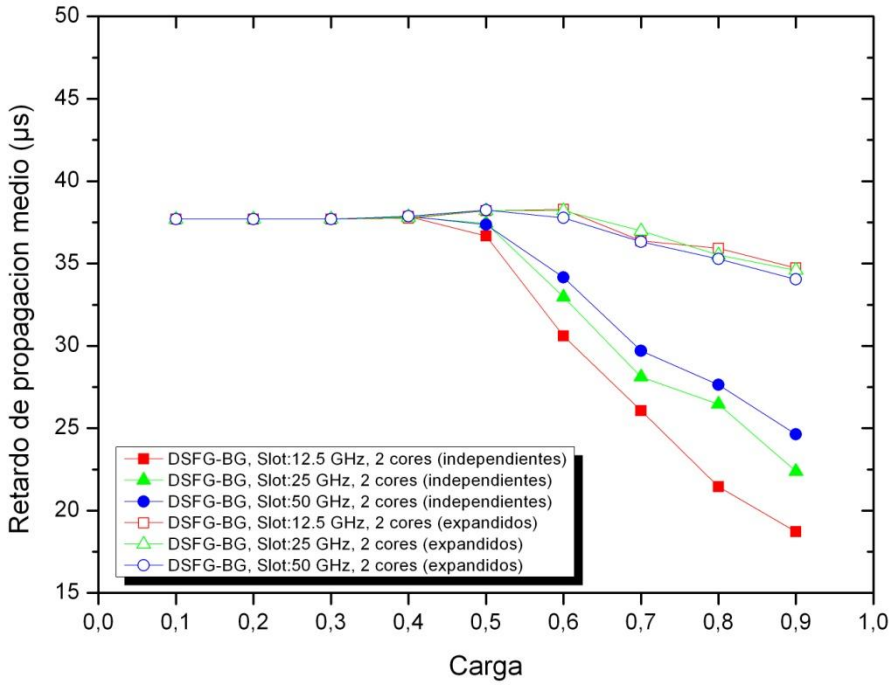

Figura 42. Red en Anillo, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

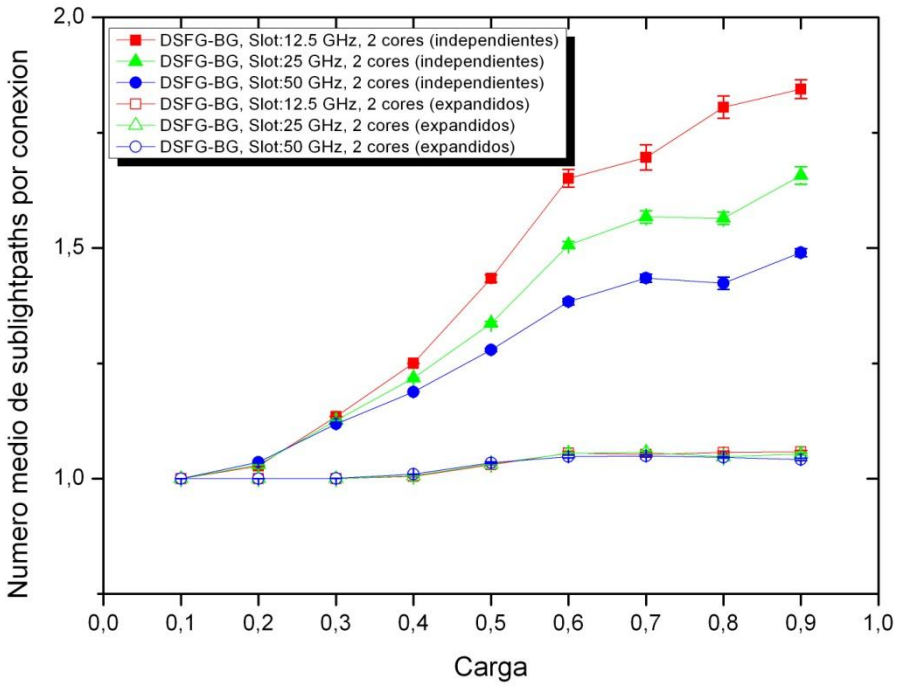

Figura 43. Red en Anillo, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En este caso la *Figura 40* muestra que la probabilidad de bloqueo ofrece mejores resultados para la configuración de 2 *cores* independientes al contrario de cómo sucedía con el algoritmo JSFG-BG, esto se explica debido a que utilizamos el criterio *best-gap* para la asignación de los huecos, entonces en la configuración de 2 *cores* independientes primero revisaríamos el *core* 1 para asignar la capacidad demandada y hasta que no se utilizara todo la capacidad disponible en el *core* 1 no pasaríamos a revisar el *core* 2 en caso de ser necesario concatenar rutas. De esta forma, se aprovechan mejor huecos pequeños (a costa de fragmentar más la demanda como se observa en la *Figura 43*.

Mientras que en el caso de 2 *cores* expandidos tenemos un único *core* con mayor capacidad, lo que implicaría que no hacemos un uso tan eficiente de la capacidad como en el caso de la configuración de 2 *cores* independientes.

Los resultados del tiempo de computación (*Figura 41*) son los esperados ya que en la configuración de 2 *cores* independientes tendríamos que realizar más iteraciones al tener más *cores*, para poder asignar la capacidad demandada, en caso de tener que concatenar rutas y como se comentó anteriormente cuanto menor es el tamaño de slot el tiempo de computación aumenta.

La *Figura 42* nos muestra los resultados del retardo de propagación medio, la configuración de 2 *cores* independientes establece *lightpaths* más cortos en cargas más altas, esto se debe a esta configuración de la red permitirá, por su mejor gestión de la capacidad de los *cores*, hacer uso en más casos de las rutas más cortas, mientas que en

el caso de 2 *cores* expandidos se hace un uso más ineficiente y se tenderá a usar rutas más largas para establecer los *lightpaths* y suplir la capacidad demandada.

El número medio de *sublightpaths* por conexión (*Figura 43*) también muestra un resultado esperado, la configuración de 2 *cores* independientes concetanará más rutas para suplir la capacidad demandada ya que tiene un mayor número de *cores* para distribuir esa capacidad de la forma más eficiente.

De esta comparativa podemos deducir que el algoritmo DSFG-BG es más eficiente en términos de probabilidad de bloqueo con la configuración de 2 *cores* independientes, al contrario de cómo sucedía en la comparativa de las configuraciones para el algoritmo JSFG-BG.

4.2.5. Comparativa entre métodos Joint y Disjoint

Como último apartado se compararán las distintas configuraciones de 2 *cores* expandidos y 2 *cores* independientes con los algoritmos *joint* y *disjoint*. Se analizarán los resultados para la probabilidad de bloqueo (*Figura 44*), tiempo de computación (*Figura 45*) y retardo de propagación medio por lightpath (*Figura 46*), con tamaño de slot de 12,5 GHz y rutas ordenadas según *shortest-path*.

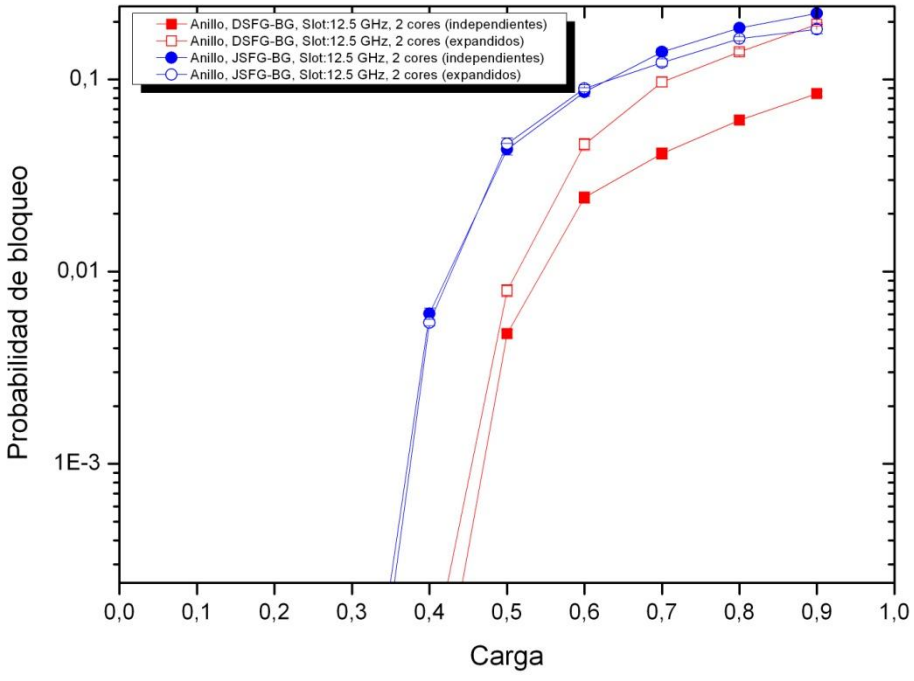

Figura 44. Red en Anillo, Probabilidad de bloqueo de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

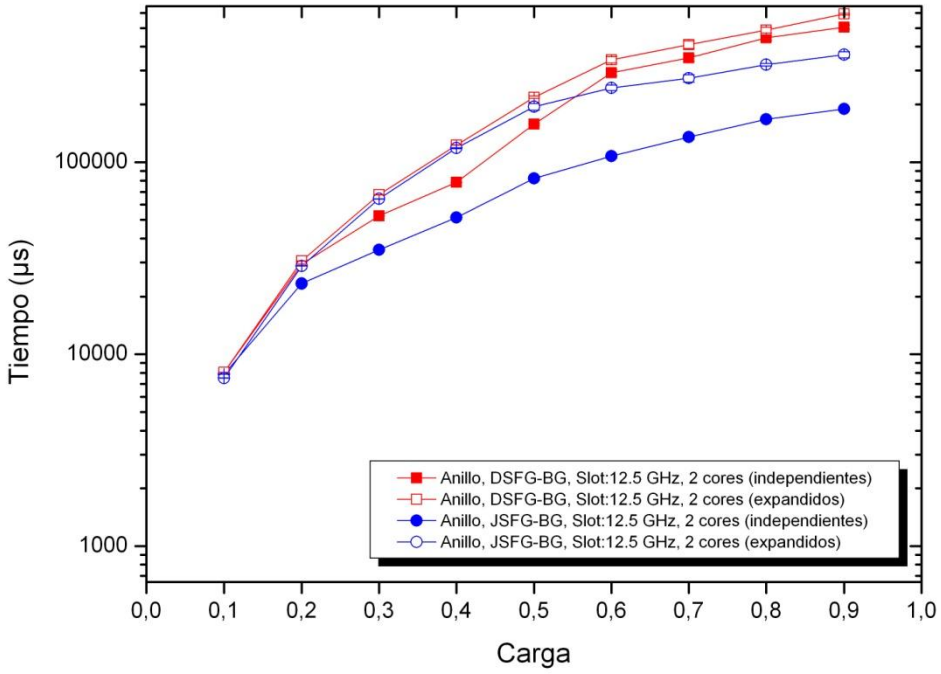

Figura 45. Red en Anillo, Tiempo de computación de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

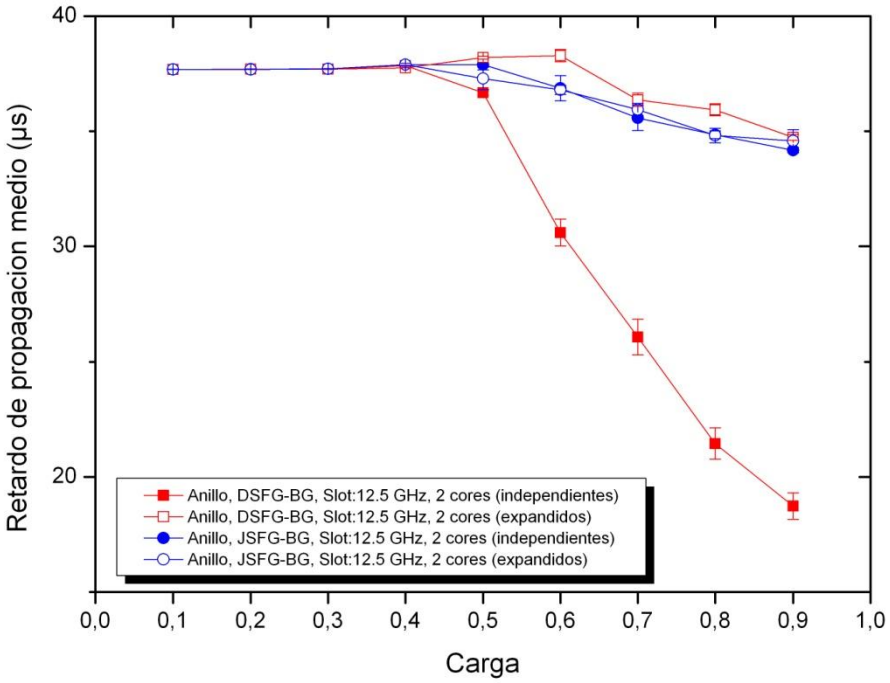

Figura 46. Red en Anillo, Retardo de propagación medio por lightpath de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Podemos observar que, en la red en anillo, los algoritmos *disjoint* ofrecen mejores resultados en la probabilidad de bloqueo (*Figura 44*) cerca de un orden de magnitud a cambio de subdividir la demanda en varios lightpaths y tener que hacer *traffic grooming*.

Si observamos el tiempo de computación (*Figura 45*), tanto el algoritmo *joint* como el *disjoint* con la configuración de 2 *cores* expandidos ofrece resultados similares, mientras que en la configuración de 2 *cores* independientes la configuración *joint* ofrece mejores resultados que *disjoint*. Esto se debe a que con los algoritmos *disjoint*, se realizan más iteracciones para la asignación de la capacidad demandada en caso de que no se encuentre un hueco donde podamos asignar toda la capacidad demandada y se tenga que fragmentar esta capacidad para poder establecer el *lightpath*.

La longitud de los *lightpaths* (*Figura 46*) es similar en todas las cargas, tanto para *joint* como para *disjoint* en ambas configuraciones, menos en cargas de 0.9 que los *lightpaths* establecidos son más cortos ya que saturaríamos antes la red y no permitiría establecer *lightpaths* más largos (en número de saltos), aunque apreciamos que la configuración de 2 *cores* independientes comienza no poder crear *lightpaths* más largos desde cargas de 0,5.

4.3. **Red NSFNET**

Analizaremos los resultados para la red NSFNet para los mismos algoritmos que en la red en anillo con los mismos parámetros, con ordenación de rutas según el criterio *k-shortest paths*, numero de rutas (*k*) de 2 y asignación de huecos con el criterio *best-gap*.

Como los resultados y las gráficas obtenidos para los algoritmos JSFG-BG y DSFG-BG y cada una de sus configuraciones ofrecen las mismas conclusiones que en la red en anillo haremos la comparativa entre las distintas configuraciones de 1 *core*, 2 *cores* expandidos y 2 *cores* independientes para cada uno de los algoritmos para ver cuál de estas es más eficiente.

Utilizaremos los tamaños de slot que han ofrecido los mejores resultados en la red en anillo, ya que ofrecen también los mejores resultados en la red NFSNet, es decir 12.5, 25 y 50 GHz. Los otros tamaños también se probaron y se llegaron a las mismas conclusiones. Por evitar repetir el mismo análisis, nos centramos en éstos.

4.3.1. Comparativa Algoritmos JSFG-BG

Comenzaremos comparando las configuraciones de 1 *core* y 2 *cores* independientes con los parámetros anteriormente mencionados para ver cual ofrece mejores resultados en la probabilidad de bloqueo (*Figura 47*), tiempo de computación (*Figura 48*) y retardo de propagación medio (*Figura 49*).

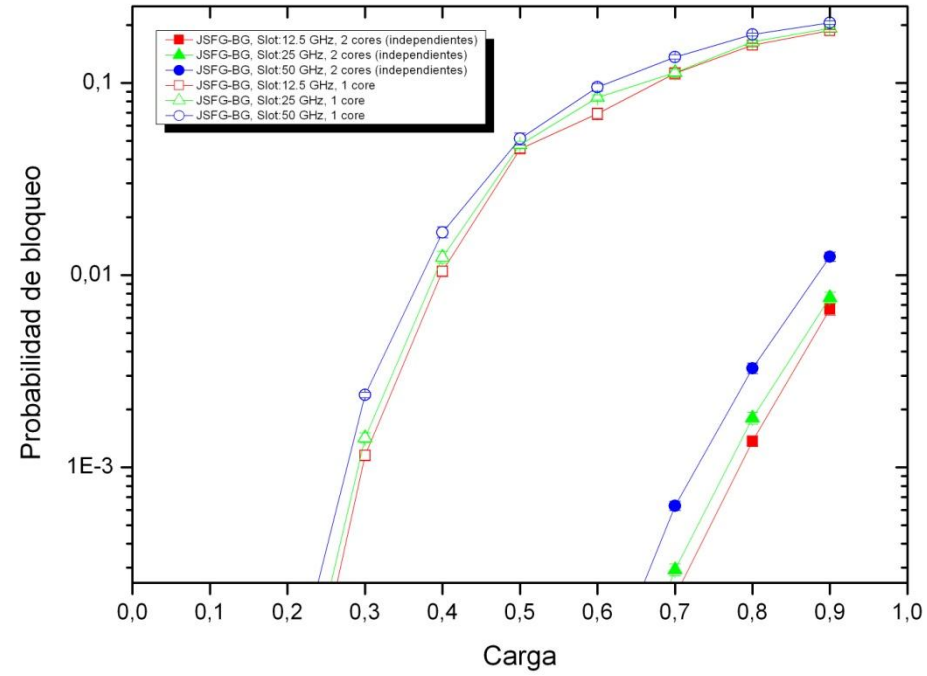

Figura 47.Red NSFNet, Probabilidad de bloqueo del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

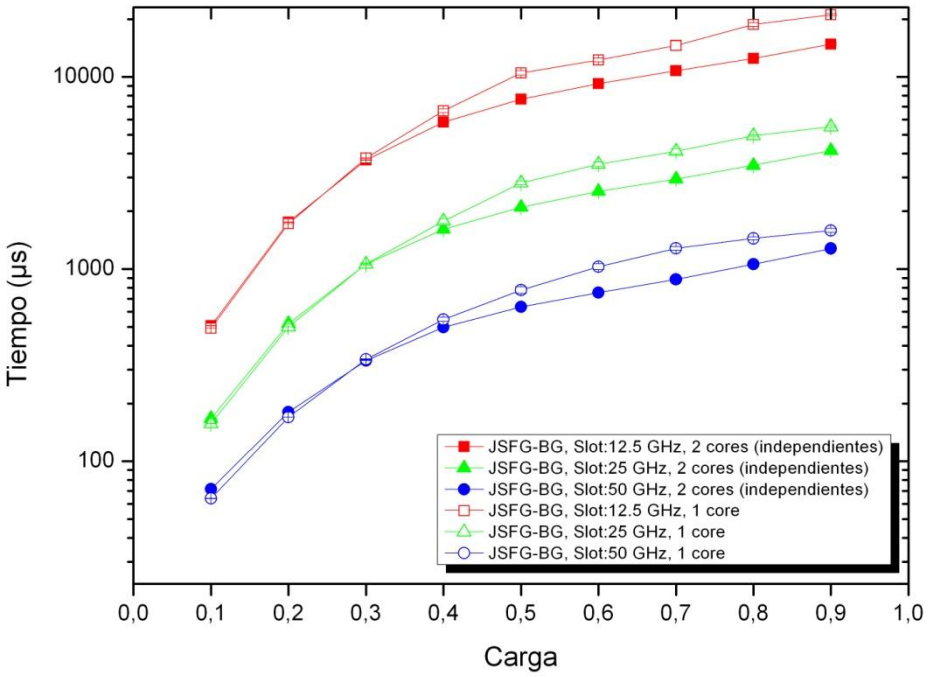

Figura 48. Red NSFNet, Tiempo de computación del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

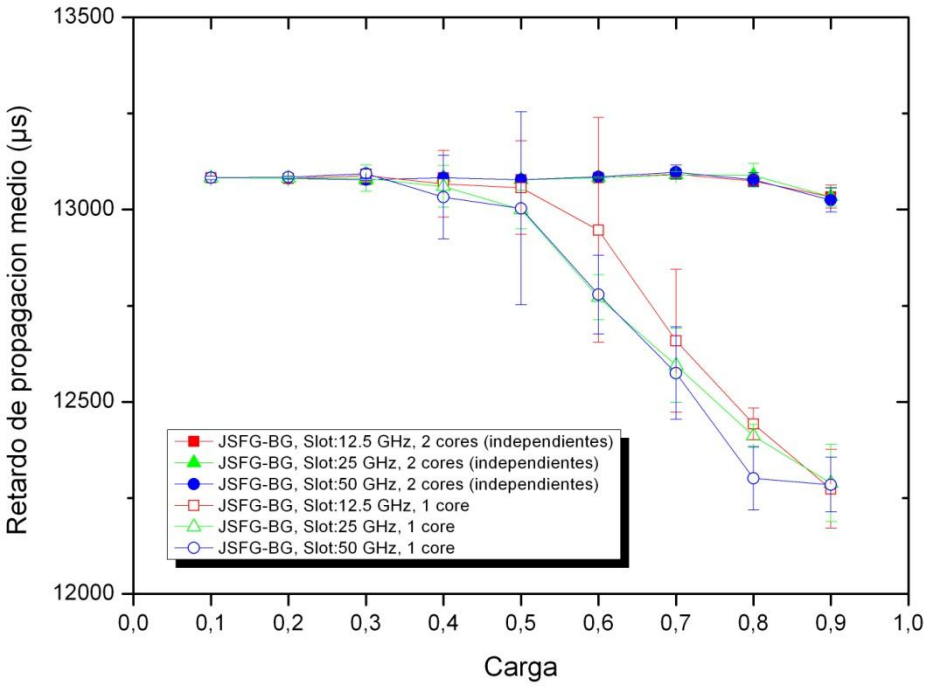

Figura 49. Red NSFNet, Retardo de propagación medio por lightpath del algoritmo JSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

En cuanto a la probabilidad de bloqueo en la *Figura 47* observamos que los resultados para la configuración de 2 *cores* independientes son mejores que para un único *core*. Para obtener probabilidades de bloqueo de 0,01 con la configuración de 1 *core* podemos trabajar con cargas de 0,4, mientras que con la configuración de 2 *cores* independientes podemos trabajar con cargas de 0,9 para obtener los mismos resultados.

Además, en la *Figura 48*, podemos observar que también ofrece mejores resultados en cuanto al tiempo de computación ya que con la configuración de 2 *cores* independientes tenemos más posibilidades para establecer los *lightpaths* por la ruta más corta y no será necesario comprobar las siguientes rutas y por lo tanto el tiempo de computación será menor.

La *Figura 49*, nos muestra que la configuración de 2 *cores* independientes permite establecer *lightpaths* de mayor longitud, ya que al disponer de una mayor capacidad permite que la longitud de los *lightpaths* (en número de saltos) sea mayor que en el caso de la configuración de 1 *core*.

La configuración que ofrece mejores resultados, como era de esperar, es la configuración de 2 *cores* independientes ya que mejora mucho la probabilidad de bloqueo, ahora compararemos esta con la configuración de 2 *cores* expandidos.

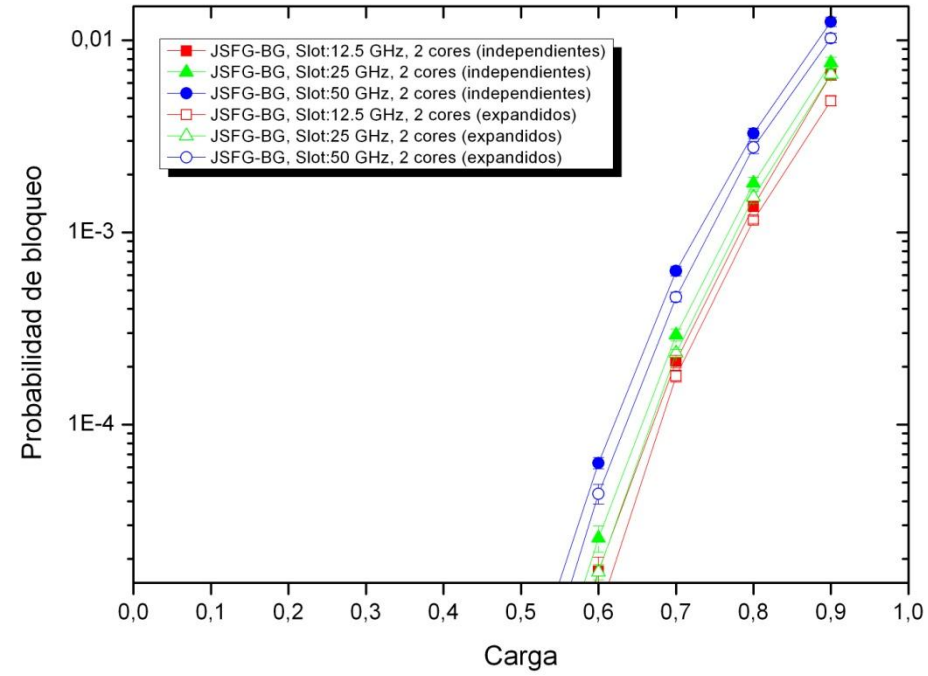

Figura 50. Red NSFNet, Probabilidad de bloqueo del algoritmo JSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

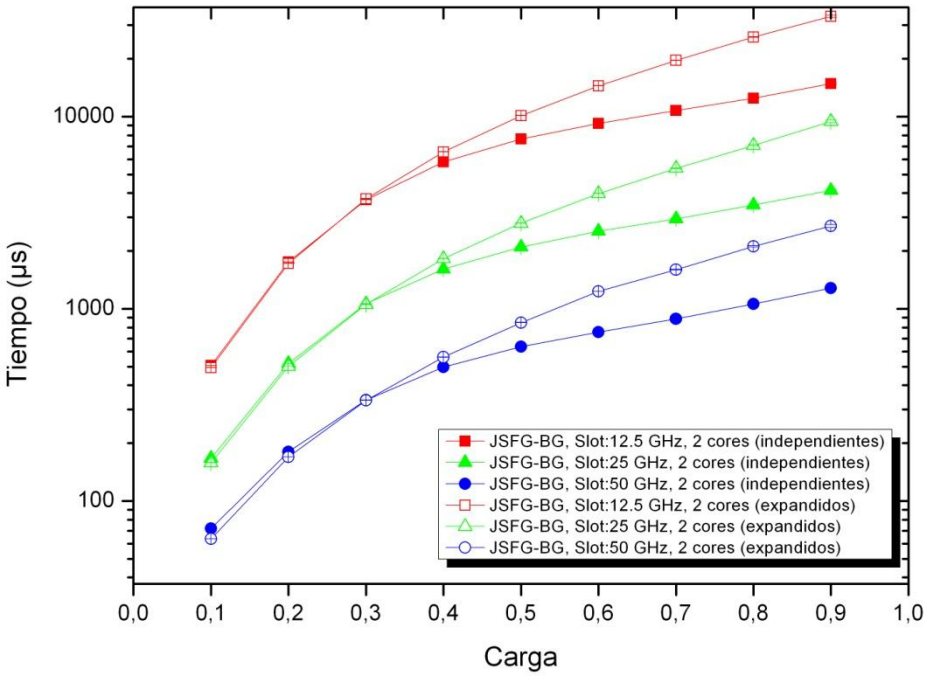

Figura 51. Red NSFNet, Tiempo de computación del algoritmo JSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Los resultados obtenidos para la probabilidad de bloqueo (*Figura 50*) son similares en ambas configuraciones pero los métodos de expectro expandido presentan un valor ligeramente menor.

En el tiempo de computación (*Figura 51*) se observan resultados similares a los obtenidos en la red en anillo, en el caso de la configuración de 2 *cores* independientes tenemos más alternativas para establecer los *lightpaths* por las rutas más cortas (en número de saltos) y por lo tanto tendremos que realizar menos iteraciones para asignar el ancho de banda demandado.

Al igual que en la red en anillo podemos deducir que la configuración más eficiente para el algoritmo JSFG-GB es la configuración de 2 *cores* expandidos si lo que nos interesase es la probabilidad de bloqueo.

4.3.2. Comparativa Algoritmos DSFG-BG

Como con los algoritmos JSFG-BG comenzaremos comparando las configuraciones de 1 *core* y 2 *cores* independientes con los parámetros mencionados anteriormente.

Analizaremos los parámetros de la probabilidad de bloqueo (*Figura 52*), tiempo de computación (*Figura 53*), retardo de propagación medio (*Figura 54*) y número de *sublightpaths* por conexión (*Figura 55*).

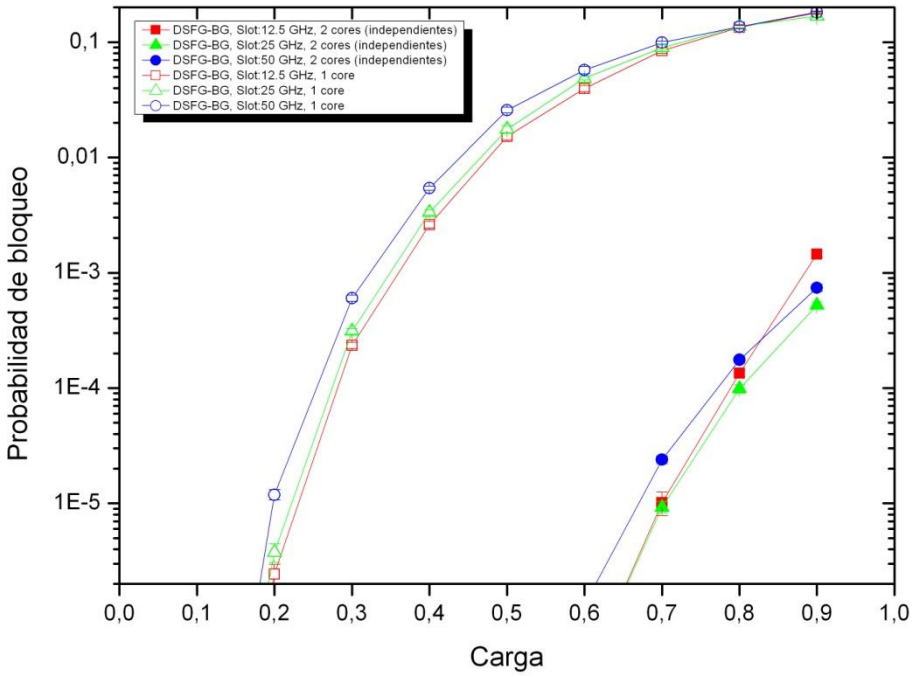

Figura 52. Red NSFNet, Probabilidad de bloqueo del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

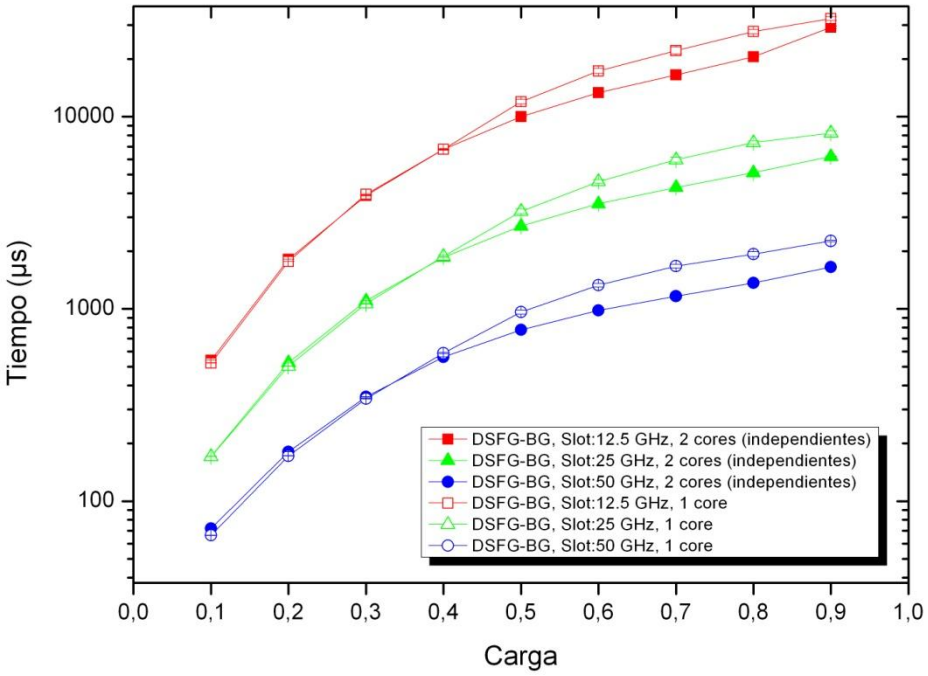

Figura 53. Red NSFNet, Tiempo de computación del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

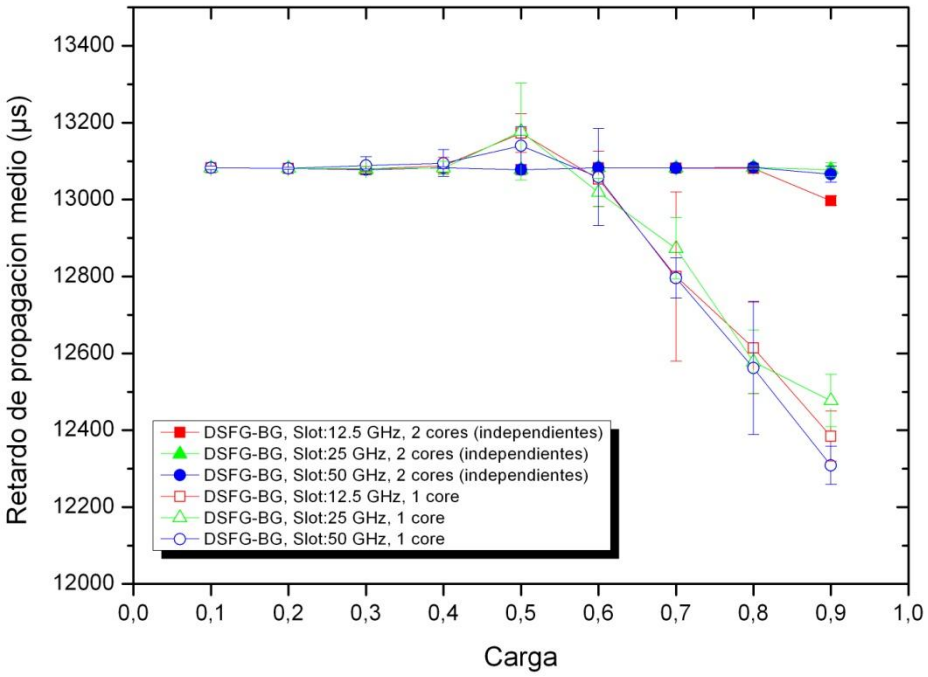

Figura 54. Red NSFNet, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

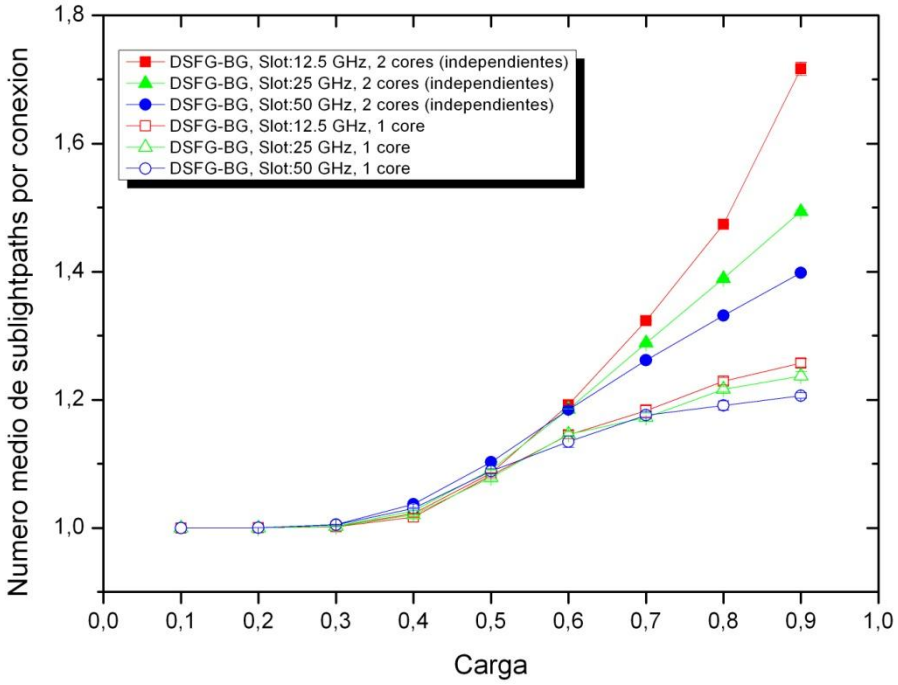

Figura 55. Red NSFNet, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (1 core y 2 cores independientes) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

La probabilidad de bloqueo (*Figura 52*) muestra los resultados esperados, la configuración de 2 *cores* independientes ofrece mejores resultados. Como podemos observar para tener probabilidades de bloqueo de 0,001 con la configuración de 1 *core* trabajamos con 0,3 de carga y con la configuración de 2 *cores* independientes trabajaríamos con cargas de 0,9.

Las conclusiones que podemos obtener del resto de gráficos son las mismas que en la comparativa de estos algoritmos para la red en anillo.

El tiempo de computación (*Figura 53*), la configuración de 2 *cores* independientes ofrece mejores tiempos de computación ya que se dispondrán de más opciones para establecer los *lightpaths* por las rutas más cortas y por lo tanto serán necesarias menos iteraciones para poder asignar toda la capacidad demandada.

En cuanto al retardo de propagación medio (*Figura 54*), la configuración de 2 *cores* independientes permite establecer *lighpaths* más largos, ya que al tener una mayor capacidad permiten que se establezcan más independientemente del número de saltos.

El número medio de *sublighpaths* por conexión (*Figura 55*) nos indica que la configuración de 2 *cores* independientes realiza un mayor número de sublightpahts por conexión que la configuración de 1 *core*, esto se debe a que en la configuración de 2 *cores* independientes hay más opciones para fragmentar la capacidad demandada ya que contamos con un mayor número de *cores*.

Por lo tanto la configuración de 2 *cores* independientes ofrece mejores resultados que la configuración de 1 *core*, como era de esperar. Ahora compararemos esta con la configuración de 2 *cores* expandidos para ver cual ofrece mejores resultados.

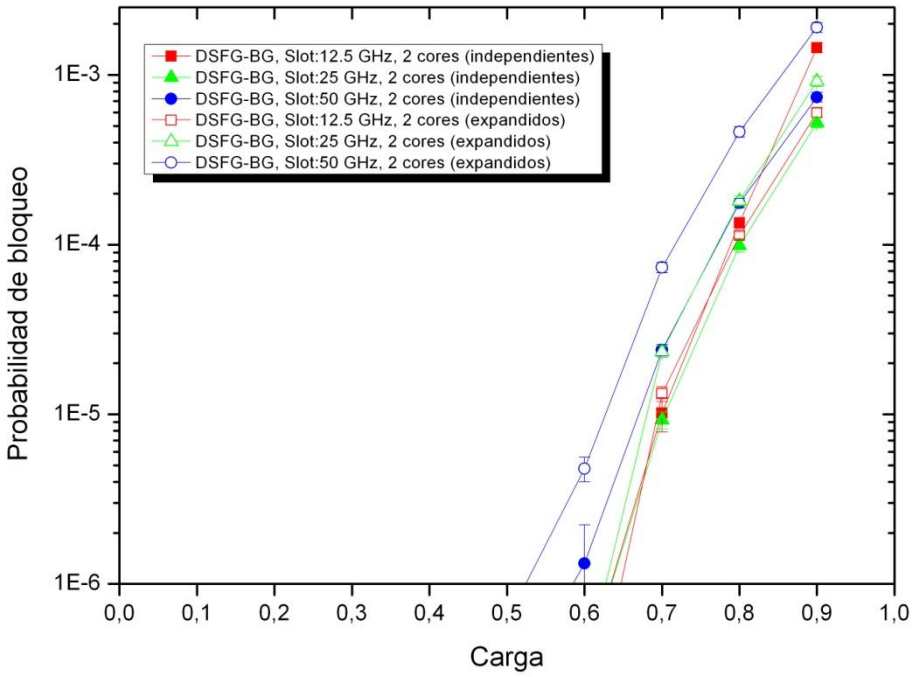

Figura 56. Red NSFNet, Probabilidad de bloqueo del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

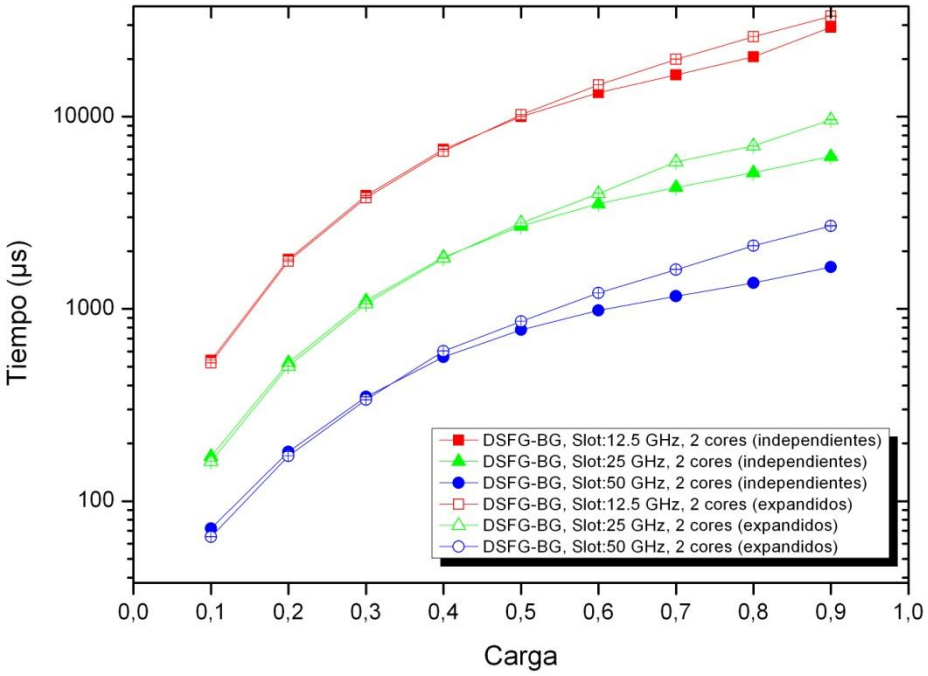

Figura 57. Red NSFNet, Tiempo de computación del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

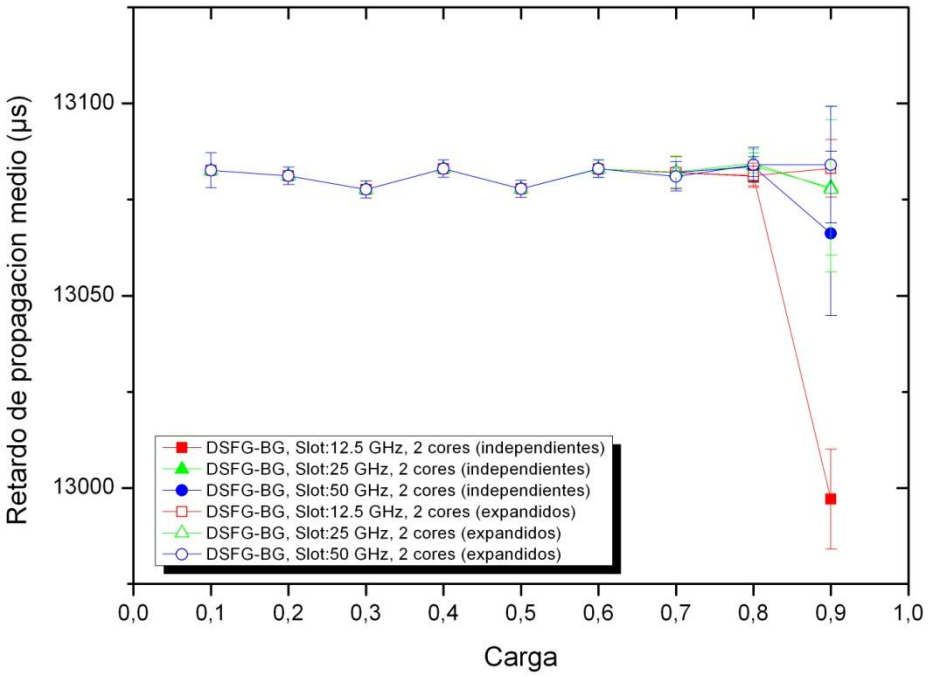

Figura 58. Red NSFNet, Retardo de propagación medio por lightpath del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

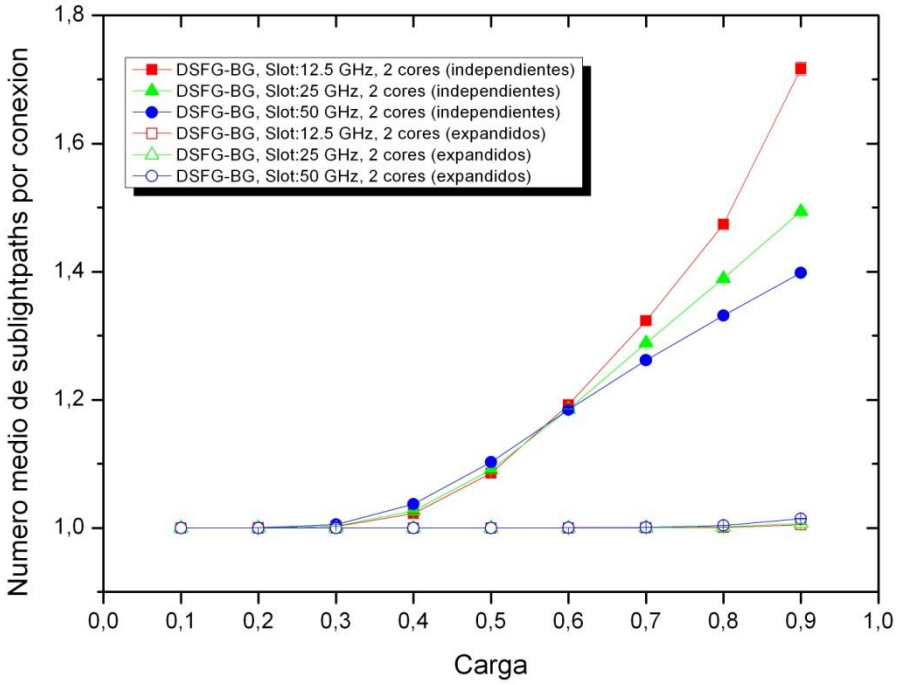

Figura 59. Red NSFNet, Número medio de sublightpaths por conexion del algoritmo DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Como ya sucedía en la red en anillo, la configuración de 2 *cores* independientes ofrece mejores resultados para la probabilidad de bloqueo (*Figura 56*), ya que asignaremos la capacidad demandada según el criterio *best-gap* en el *core* 1 y hasta que no se haya ocupado toda la capacidad de este, no pasaremos a revisar el *core* 2, por lo tanto se aprovechará mejor la capacidad de los *cores*, ya que hasta que no tengamos uno completo no pasaremos al siguiente. Mientras que en la configuración de 2 *cores* expandidos es como si tenemos un único *core* con más capacidad por lo tanto haremos un peor uso de la capacidad del mismo porque no aprovechará los huecos. Por el contrario, como se ve en la *Figura 59*, el método de cores independientes fragmenta más la demanda en varios sublightpaths.

El tiempo de computación (*Figura 57*) nos muestra los mismos resultados que en la red en anillo, cuanto menor es el tamaño de slot mayor es el tiempo de computación como, como era de esperar. La configuración de 2 *cores* independientes ofrece mejores tiempos de computación, ya que habrá disponibles más posibilidades para establecer los *lightpaths* por las rutas más cortas y por lo tanto se realizarán menos iteracciones para suplir la capacidad demandada. Mientras que en la configuración de 2 *cores* expandidos el número de *cores* es menor y por lo tanto tendrá que recurrir en más ocasiones a rutas más largas.

Los resultados del retardo de propagación medio (*Figura 58*) muestran que ambas configuraciones ofrecen los mismos resultados y permiten establecer *lightpaths* de la misma longitud (en número de saltos) en todas las cargas.

El número medio de *sublightpaths* por conexión (*Figura 59*) indica que la configuración de 2 *cores* independientes fragmenta mucho más la capacidad demandada ya que tiene más opciones disponibles para establecer los *lightpaths* al disponer de un mayor número de *cores*.

Visto esto, podemos ver que la configuración más eficiente para el algoritmo DSFG-BG en la red NFSNet, es la configuración de 2 *cores* independientes.

4.3.3. Comparativa entre métodos Joint y Disjoint

Se compararán las distintas configuraciones de 2 *cores* expandidos y 2 *cores* independientes con los algoritmos *joint* y *disjoint*. Se analizarán los resultados para la probabilidad de bloqueo (*Figura 60*), tiempo de computación (*Figura 61*) y retardo de propagación medio por lightpath (*Figura 62*), con tamaño de slot de 12,5 GHz y rutas ordenadas según *shortest-path*.

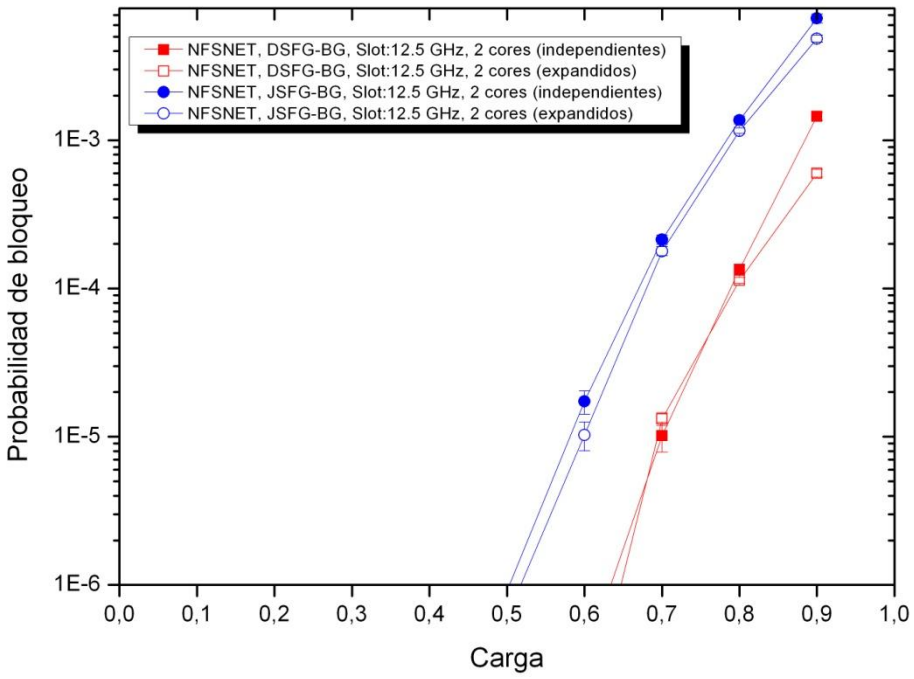

Figura 60. Red NFSNet, Probabilidad de bloqueo de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

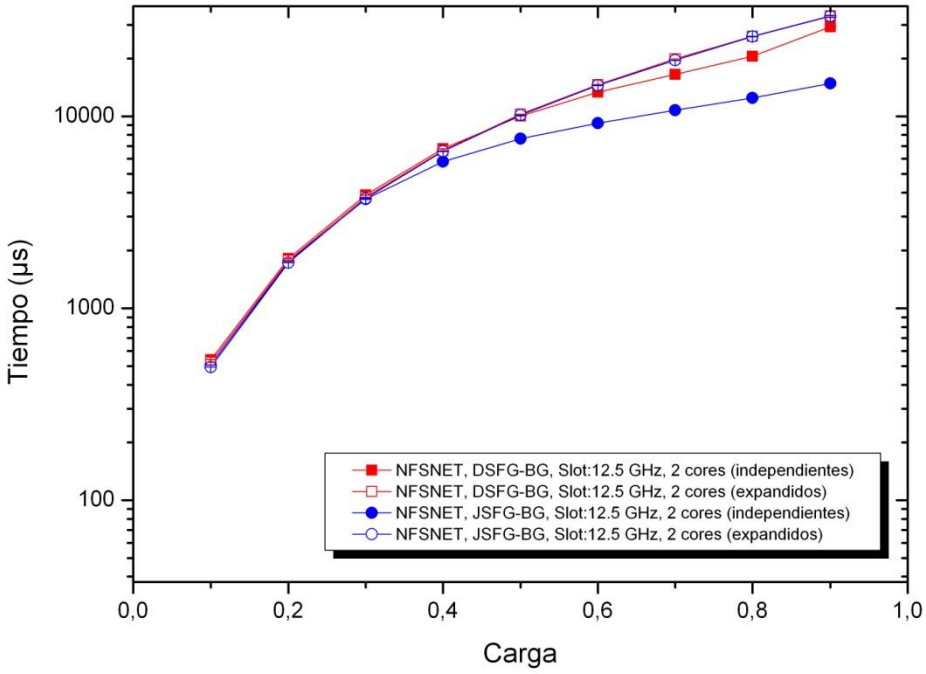

Figura 61. Red NFSNet, Tiempo de computación de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

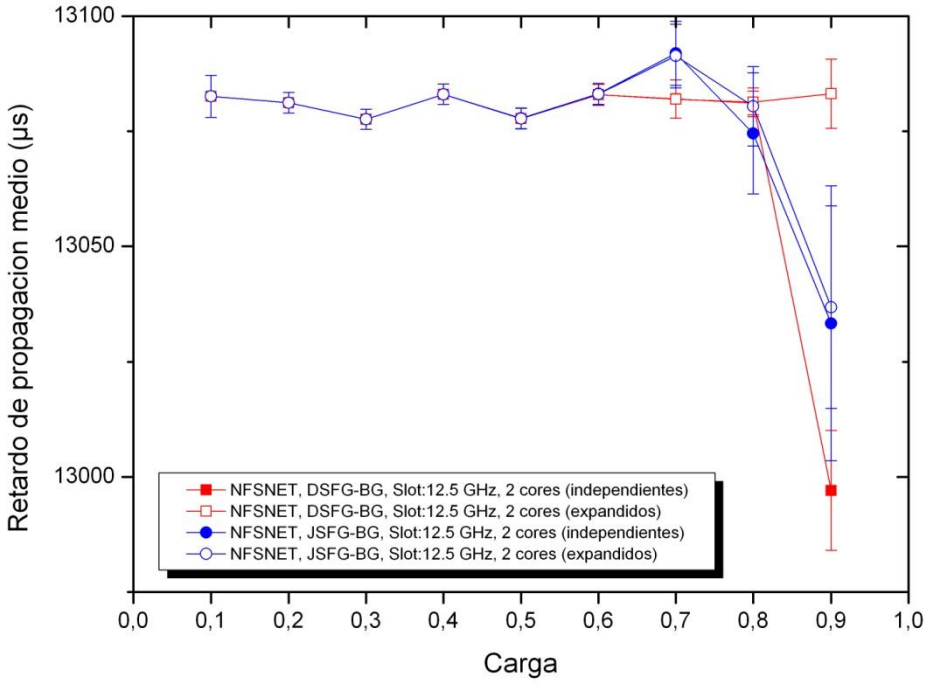

Figura 62. Red NFSNet, Retardo de propagación medio por lightpath de los algoritmos JSFG-BG y DSFG-BG (2 cores independientes y 2 cores expandidos) para varios tamaños de slot, con k=2, ordenación de rutas según shortest-path y barriendo la carga.

Podemos observar, como ya ocurría en la red en anillo, que en la red NFSNet los algoritmos *disjoint* ofrecen mejores resultados en la probabilidad de bloqueo (*Figura 60*) cerca de un orden de magnitud a cambio de subdividir la demanda en varios lightpaths y tener que hacer *traffic grooming*.

En cuanto al tiempo de computación (*Figura 61*), tanto el algoritmo *joint* como el *disjoint* con la configuración de 2 *cores* expandidos ofrece resultados similares, mientras que en la configuración de 2 *cores* independientes la configuración *joint* ofrece mejores resultados que *disjoint*. Esto se debe a que con los algoritmos *disjoint*, se realizan más iteracciones para la asignación de la capacidad demandada en caso de que no se encuentre un hueco donde podamos asignar toda la capacidad demandada y se tenga que fragmentar esta capacidad para poder establecer el *lightpath*

El retardo de propagación medio por *lightpath* en la red NFSNet (*Figura 62*), la longitud de los *lightpaths* es similar en todas las cargas menos en cargas de 0.9 que los lightpaths establecidos son más cortos ya que saturaríamos antes la red y no permitiría establecer *lightpaths* más largos (en número de saltos).

Capítulo 5. Conclusiones y líneas futuras

En este TFG se ha adaptado el simulador de redes ópticas elásticas que actualmente posee el Grupo de Comunicaciones Ópticas de la Universidad de Valladolid para que pueda trabajar con redes con fibra *multicore*. A la hora de convertir los métodos que actualmente tiene al grupo, se han propuesto dos alternativas para las redes *multicore*: *cores* independientes y *cores* expandidos.

En la memoria, tras describir todo el trabajo realizado, se ha presentado un estudio para comparar el rendimiento de ambas propuestas en dos redes (anillo metropolitano y NSFNet). También se ha comparado con los resulados de una red con fibra óptica convencional de un único *core*. Todas estas pruebas se han realizado en el simulador implementado en este TFG en OMNeT++ y se ha comparado la probabilidad de bloqueo, tiempo de computación, retardo de propagación medio y número de *sublightpaths* por conexión. El más relevante de todos es la probabilidad de bloqueo y el que marcará cual es el más eficiente ya que lo que buscamos es atender el número máximo de peticiones posibles.

Las configuraciones que se han introducido variando el número de *cores* hace que el uso de fibras *multicore* permite mejores resultados en la probabilidad de bloqueo en todos los casos y en algunos hasta mejores tiempos de computación, aunque no serán todo ventajas, la utilización de fibras *multicore* supone aumentar los recursos asignados a la red por lo que hay que llegar a un compromiso entre las mejoras en las prestaciones a cambio del aumento de los recursos asignados a la misma.

Viendo los resultados obtenidos podemos decir, diferenciando las dos redes utilizadas, que en la red en anillo lo que ofrece mejores resultados en la probabilidad de bloqueo es el algoritmo DSFG-BG con la configuración de 2 *cores* independientes. Mientras que en la red NFSNet el más eficiente es el algoritmo DSFG-BG con la configuración de 2 *cores* expandidos.

Como líneas futuras, se podría:

- Hacer pruebas con un mayor número de *cores* para ambas configuraciones planteadas en este TFG y observar si ofrecen buenos resultados. En este TFG no se han incluido, aunque están en proceso de simulación en una de las máquinas usadas para las simulaciones ya que el tiempo para la obtención de resultados es demasiado largo.
- Comprobación de estos algoritmos con *first-fit* en lugar de con *best-gap* y una comparativa entre ambos criterios para ver cual ofrece mejores resultados.
- Introducir un método de protección para estas configuraciones *multicore*.
- Propuesta de nuevos métodos en redes estáticas diseñando la topología virtual pero utilizando fibras multicore.
- Diseñar algoritmos que tengan en cuenta en impacto del *crosstalk* a la hora de realizar la asignación de recursos.

Referencias

[1] Telecomunicaciones TICS, https://telecomunicaciones2.webnode.mx/unidad-6/

[2] R. Ramaswami, "Optical Fiber Communication: From Transmission To Networking", IEEE Communications Magazine, vol. 40, no. 5, pp. 138-147, May 2002.

[3] B. Mukherjee, "WDM Optical Communication Networks: Progress and Challenges", IEEE Journal on Selected Areas in Communications, vol. 18, no. 10, pp. 1810-1824, October 2000.

[4] R. Ramaswami, K.N. Sivarajan, and G.H. Sasaki, *Optical Networks: A Practical Perspective*, Third Edition, Morgan Kaufmann, 2010.

[5] K. Lu, G. Xiao, and I. Chlamtac, "Analysis of Blocking Probability for Distributed Lightpath Establishment in WDM Optical Networks", IEEE/ACM Transactions on Networking, vol. 13, no. 1, pp. 187-197, February 2005.

[6] B.C. Chatterjee, N. Sarma, and E. Oki, "Routing and Spectrum Allocation in Elastic Optical Networks: A Tutorial", IEEE Communications Surveys & Tutorials, vol. 17, no. 3, pp. 1776-1800, Third Quarter 2015.

[7] J.J. Granada-Torres, A. M. Cárdenas-Soto, N. Guerrero-González, "Redes ópticas elásticas: un nuevo paradigma en las futuras redes de telecomunicaciones", *Respuestas*, vol. 20, no. 2, pp. 6-22, 2015.

[8] I. Tomkos, S. Azodolmolky, J. Solé-Pareta, D. Careglio, and E. Palkopoulou, "A Tutorial on the Flexible Optical Networking Paradigm: State of the Art, Trends, and Research Challenges", Proceedings of the IEEE, vol. 102, no. 9, pp. 1317-1337, September 2014.

[9] K. Christodoulopoulos, I. Tomkos, and E.A. Varvarigos, "Routing and Spectrum Allocation in OFDM-based Optical Networks with Elastic Bandwidth Allocation", 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, 2010.

[10] B.C. Chatterjee, N. Sarma, and E. Oki, "Routing and Spectrum Allocation in Elastic Optical Networks: A Tutorial", IEEE Communications Surveys & Tutorials, vol. 17, no. 3, pp. 1776-1800, Third Quarter 2015.

[11] D. Klonidis, F. Cugini, O. Gerstel, M. Jinno, V. López, E. Palkopoulou, M. Sekiya, D. Siracusa, G. Thouénon, and C. Betoule, "Spectrally and Spatially Flexible Optical Network Planning and Operations", IEEE Communications Magazine, vol. 53, no. 2, pp. 69-78, February 2015.

[12] Sara Fernández Atienza, "Dimensionado y análisis tecno-económico de redes ópticas elásticas", *Trabajo Fin de Master*, Master en Ingeniería de Telecomunicación, E.T.S.I de Telecomuniación, Universidad de Valladolid,

2016(https://uvadoc.uva.es/bitstream/handle/10324/21028/TFM-G661.pdf?sequence=1).

[13] Andrés Macho Ortiz, “Multi-core fiber and Optical Supersymmetry: Theory and Applications”, *Tesis Doctoral*, Universitat Politécnica de Valéncia, 2019 (https://riunet.upv.es/handle/10251/124964).

[14] Pablo Jesús López Méndez, “Diseño de un modelo de simulación para la capacidad de enalces mediante multiplexación de nodos”, *Trabajo Fin de Master*, Master en Ingeniería de Telecomunicación, Universitat Oberta de Catalunya, 2019(http://openaccess.uoc.edu/webapps/o2/bitstream/10609/90147/8/plopezmenTFM0119memoria.pdf).

[15] Illán González Horna, “Evaluación de algoritmos de asignación de recursos que ofrezcan protección en redes ópticas elásticas”, *Trabajo de Fin de Grado*, Grado en Ingeniería de Tecnologías de Telecomunicación, E.T.S.I de Telecomunicación, Universidad de Valladolid, 2017 (http://uvadoc.uva.es/handle/10324/27579)

[16] H. Tode and Y. Hirota, "Routing, Spectrum, and core and/or mode assignment on space-division multiplexing optical networks [invited]," in *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 1, pp. A99-A113, Jan. 2017, doi: 10.1364/JOCN.9.000A99.

[17] Rumipamba-Zambrano, Rubén & Moreno-Muro, Francisco-Javier & Perelló, Jordi & Pavon-Marino, Pablo & Spadaro, Salvatore. (2018). Space Continuity Constraint in Dynamic Flex-Grid/SDM Optical Core Networks: An Evaluation with Spatial and Spectral Super-channels. Computer Communications. 126. 10.1016/j.comcom.2018.05.013.

[18] Ghose, Sujoy & Kumar, Rajeev & Banerjee, Nilanjan & Datta, Raja. (2005). Multihop Virtual Topology Design in WDM Optical Networks for Self-Similar Traffic. Photonic Network Communication. 10. 199-214. 10.1007/s11107-005-2484-2.