UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA

DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Evaluación de algoritmos de asignación de recursos que ofrezcan protección en redes ópticas elásticas

Autor:

D. Illán González Horna

Tutores:

Dr. D. Ramón J. Durán Barroso

Dr. D. Ignacio de Miguel Jiménez

Valladolid, 4 de Julio de 2017

TÍTULO:	Evaluación de algoritmos de asignación de recursos que ofrezcan protección en redes ópticas elásticas
AUTOR:	D. Illán González Horna
TUTORES:	Dr. D. Ramón J. Durán Barroso Dr. D. Ignacio de Miguel Jiménez
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Dr. D. Ignacio de Miguel Jiménez
VOCAL:	Dr. D. Ramón J. Durán Barroso
SECRETARIO:	Dr. Dña Noemí Merayo Álvarez
SUPLENTE:	Dr. D. Juan Carlos Aguado Manzano
SUPLENTE:	Dr. D. Rubén M. Lorenzo Toledo
FECHA:	4 de Julio de 2017
CALIFICACIÓN:	

Resumen de TFG

Las redes ópticas convencionales con encaminamiento por longitud de onda (WRON) no pueden satisfacer la continua y creciente demanda de ancho de banda. Para solucionar este problema, la comunidad científica ha optado por dotar de flexibilidad a estas redes, de modo que las redes se adapten a las peticiones de conexión de los usuarios, redes conocidas como WRON elásticas que están basadas en conmutación de circuitos (lightpaths) y tienen que resolver el problema de encaminamiento y asignación de espectro (RSA). En este Trabajo de Fin de Grado se proponen un conjunto de algoritmos para resolver el problema RSA en redes ópticas con protección y sin ella. Además, se ha realizado un estudio mediante un simulador desarrollado en OMNeT++, para comprobar la eficacia de cada algoritmo en términos de probabilidad de bloqueo y tiempos de computación.

Palabras clave

WRON elásticas, protección, OMNeT++, algoritmos de asignación de recursos, encaminamiento y asignación espectral, probabilidad de bloqueo, tiempo de computación, eficiencia.

Abstract

Conventional Wavelength-Routed Optical Networks (WRON) can no longer keep up with the emerging bandwidth-consuming services, such as cloud computing or high definition TV. In order to solve this problem, the research community is moving towards flexible networks which are able to adjust their resources according to the requirements of each connection. These networks, called elastic WRONs, are based on the establishment of lightpaths (optical circuits) and they need efficient design methods that can solve the Routing and Spectrum Assignment (RSA) for each lightpath. In this work, we have proposed and implemented a set of algorithms to solve that problem in both protected and unprotected elastic optical networks. Furthermore, we explain the implementation of each proposal and present a simulation study about the efficiency of each algorithm, in terms of both the blocking probability and computation time. To this end, we use the discrete event simulator OMNeT++.

Keywords

Elastic WRON, protection, OMNeT++, resource allocation algorithms, routing and spectrum allocation, blocking probability, computation time, efficiency.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Las comunicaciones ópticas son, a día de hoy, la mejor forma de transmitir grandes cantidades de datos en largas distancias con poca latencia [1]. En comparación con otros medios de transmisión, como el cable coaxial o el par trenzado, la fibra óptica ofrece un medio de transmisión cuasi perfecto: pocas pérdidas sobre un gran ancho de banda, baja atenuación y distorsión de la señal, inmunidad ante interferencias electromagnéticas, bajos requisitos de potencia, pequeños requerimientos de espacio, gran durabilidad y bajo coste [2, 3].

Desde sus inicios en la década de 1970, la capacidad de transmisión de la tecnología de redes ópticas ha avanzado de manera extraordinaria. Los primeros años de las comunicaciones ópticas estuvieron marcados por el rápido desarrollo de la tecnología, un aumento moderado de la tasa de transmisión y el cambio gradual de ventana de transmisión [1]. Poco después apareció la primera generación de redes ópticas. En ella, la óptica se utiliza esencialmente para transmitir y proporcionar capacidad, mientras que la electrónica se encarga de toda la conmutación y otras funciones inteligentes. Ejemplos de redes ópticas de primera generación son SDH y SONET [4].

A principios de la década de 1990, se inventaron los EDFAs (*Erbium Doped Fiber Amplifier*), cuyo factor determinante es poder amplificar señales en varias longitudes de onda simultáneamente. Esto permitió la aparición de una nueva forma de incrementar las capacidades de los sistemas: en lugar de aumentar la tasa de transmisión (que electrónicamente tiene sus límites), se pueden transmitir más canales centrados en distintas longitudes de onda, esto es, usar WDM (*Wavelength Division Multiplexing*). El uso de WDM redujo radicalmente los costes de sistemas de transmisión de larga distancia y aumentó su capacidad [2].

Como consecuencia de estos avances, a finales de esa misma década se impulsó el despliegue de la segunda generación de redes ópticas, en la cual la capa óptica cuenta con funciones de encaminamiento, conmutación e inteligencia. En una red WDM, las conexiones todo-ópticas extremo a extremo, llamadas *lightpaths*, se establecen entre un nodo origen y un nodo destino de la red. Normalmente atraviesan nodos intermedios, donde son enrutadas y conmutadas de un enlace a otro. La principal ventaja de la comunicación mediante *lightpaths* es que proporcionan una comunicación transparente de datos y eliminan el coste y los cuellos de botella de procesamiento electrónico en los nodos intermedios. Las redes WDM basadas en *lightpaths* son conocidas como WRONs (*Wavelength-Routed Optical Network*) o redes ópticas con encaminamiento por longitud de onda [4, 5].

A pesar de estos grandes avances, la reciente aparición de servicios muy dinámicos y que requieren de gran ancho de banda, ha provocado que las WRON convencionales no puedan satisfacer semejante demanda de ancho de banda. Esto es debido a que las redes tradicionales basadas en WDM ofrecen la posibilidad de establecer conexiones rígidas (longitudes de onda) a una tasa de transmisión fija, donde los canales están modulados con un formato de modulación común (OOK,

DPSK o QPSK) y están espaciados a una distancia fija de 50 GHz. La flexibilidad en estas redes está restringida a lo permitido por los láseres sintonizables en frecuencia y a las reconfiguraciones limitadas permitidas por los conmutadores ópticos. En consecuencia, la adaptación de la red a tráfico y condiciones cambiantes supone un reto. Además, está creciendo la concienciación de que la ocupación del espectro disponible en fibras ópticas está alcanzando su límite máximo. En este marco ha emergido el paradigma de redes ópticas elásticas como un medio para ofrecer un uso eficiente de los recursos ópticos disponibles. El término "elástico" se refiere a la capacidad de la red para adaptar dinámicamente sus recursos, como el ancho de banda óptico y el formato de modulación, de acuerdo con los requisitos de cada conexión. Puesto que los algoritmos de planificación convencionales de las redes WDM no se pueden utilizar en redes elásticas, se está investigando en nuevos algoritmos de asignación de recursos [6].

Otro aspecto importante en la evolución de las redes ópticas es la supervivencia, es decir, la capacidad de la red para proporcionar un servicio continuado en presencia de fallos [7]. Con el gran volumen de tráfico transportado por las redes ópticas, fallos en la red pueden suponer pérdidas de gran cantidad de datos e ingresos. Por ello, es importante mantener una fiabilidad alta para dar soporte al creciente número de aplicaciones críticas que utilizan estas redes [8]. En la mayoría de casos, los fallos son provocados por errores humanos, siendo el fallo más habitual el corte de una fibra. Otro posible fallo es aquel que se produce en componentes activos del equipamiento de la red, como transmisores, receptores o controladores. También pueden fallar los nodos de la red, normalmente debido a catástrofes como inundaciones, incendios o terremotos. Por otro lado, hay más posibles aplicaciones de los esquemas de protección, aparte de la supervivencia, como la realización de mantenimiento en la red [4].

Existen varias formas para asegurar la supervivencia en una red óptica. Las arquitecturas de supervivencia de redes están basadas en, o bien dedicar recursos de respaldo de antemano, o bien en la restauración dinámica. En la dedicada (por ejemplo, APS, *Automatic Protection Switching*), el servicio de red interrumpido se restablece utilizando los recursos dedicados. En la restauración dinámica, se utiliza la capacidad sobrante disponible de la red. Normalmente, los esquemas de restauración dinámica son más eficientes en la utilización de la capacidad, mientras que la restauración dedicada ofrece menores tiempos de restauración y garantiza la restauración [3].

Otro asunto recurrente durante las últimas décadas en la comunidad científica vuelve a ser de actualidad es la arquitectura centralizada frente a la distribuida. Por norma general, los sistemas centralizados son más eficientes, mientras que los distribuidos son escalables. Hace una década, existía una tendencia hacia el desarrollo de sistemas distribuidos, principalmente debido al rápido crecimiento de las redes ópticas, al que los sistemas centralizados no se podían adaptar por ser poco escalables [2, 5].

Sin embargo, en los últimos años ha surgido una tecnología que ha invertido esta tendencia, SDN (*Software Defined Network*). Sus tres características principales son

la separación del plano de control del plano de datos, la centralización lógica del control y la programabilidad de funciones de red. Sus beneficios son una gestión simple de la red, rápido aprovisionamiento de recursos, buena escalabilidad y capacidad de desarrollo abierto [9, 10].

El Grupo de Comunicaciones Ópticas de la Universidad de Valladolid ha desarrollado en el simulador de eventos discretos OMNeT++ [11] un simulador de redes WRON elásticas, con el plano de control separado del plano de datos y con un control centralizado. El objetivo de este TFG es adaptar el simulador actual para permitir que ofrezca protección de recursos y, posteriormente, desarrollar nuevos algoritmos de asignación de recursos y comprobar su eficiencia, medida principalmente a través de la probabilidad de bloqueo y del tiempo de computación de los algoritmos. Estos algoritmos se pueden dividir fundamentalmente en dos tipos: elásticos (*gridless*) y rígidos (*fixed-grid*). Se explicarán las ventajas e inconvenientes de cada uno de los tipos, y se obtendrán resultados que determinarán cuál es más eficiente. También se llevará a cabo un análisis del esquema de protección implementado, y se comparará su eficiencia con respecto a los algoritmos que no usan este esquema de protección. Existen otras variables que determinarán la forma de programación de cada algoritmo que se verán a lo largo del documento, como el método de asignación de ancho de banda o el hecho de que las peticiones se puedan fraccionar en peticiones más pequeñas.

Este TFG está estructurado como sigue: el capítulo 2 documenta el estado actual de la investigación acerca de las redes ópticas elásticas con encaminamiento por longitud de onda; el capítulo 3 describe los algoritmos de asignación de recursos implementados en la herramienta OMNeT++; el capítulo 4 analiza los resultados obtenidos para cada uno de los algoritmos y realiza una comparativa; por último, el capítulo 5 concluye el trabajo.

Capítulo 2. Estado del arte

Las redes WRON convencionales son objeto de investigación desde hace más de una década. Tal y como se muestra en la Figura 1, estas redes están formadas por un tejido de conmutadores ópticos, constituido por conmutadores activos conectados por enlaces de fibra, formando una topología física arbitraria. Cada usuario final se conecta a un conmutador activo a través de un enlace, lo que se conoce como nodo de la red. La estación de acceso de cada nodo está equipada con transmisores y receptores, que pueden ser sintonizables en longitud de onda. El mecanismo básico de comunicación en una WRON son los *lightpaths*. Los nodos intermedios del camino encaminan el *lightpath* en el dominio óptico usando sus conmutadores activos, mientras que los nodos finales acceden al *lightpath* con transmisores y receptores sintonizados en la longitud de onda a la que el *lightpath* opera. En ausencia de convertidores de longitud de onda, el *lightpath* debe permanecer en la misma longitud de onda durante todo su camino en la red. El requisito fundamental en una WRON consiste en que dos o más *lightpaths* que atraviesan el mismo enlace deben estar en diferentes canales de longitud de onda para que no se produzcan interferencias entre ellos [3].

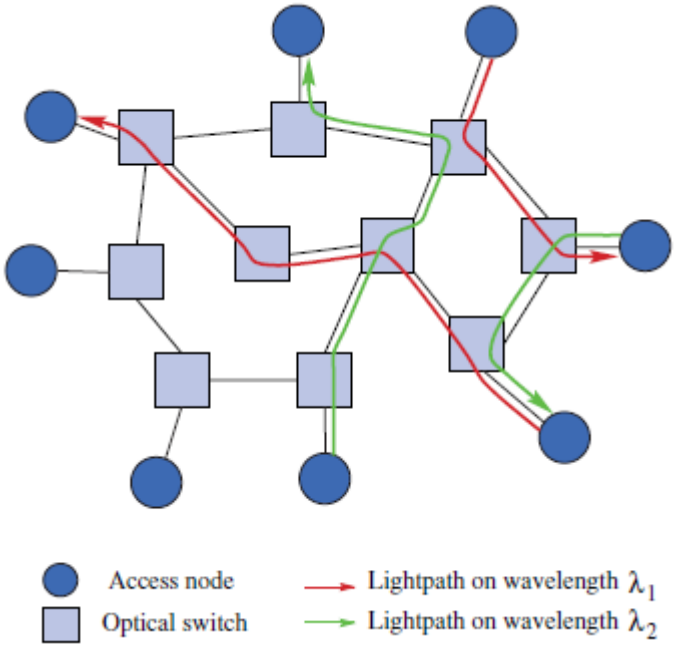


Figura 1. Ejemplo de red WRON.

En una red WRON dinámica, las peticiones de establecimiento de *lightpath* llegan en tiempo real y hay que solucionar el problema RWA (*Routing and Wavelength Assignment*) para cada una de ellas: dada una petición de establecimiento de un *lightpath*, el problema RWA consiste en determinar las rutas sobre las cuales este *lightpath* debería ser establecido y las longitudes de onda que deberían asignarse a dicho *lightpath*. Aquellos que no pueden ser establecidos debido a las restricciones en rutas y longitudes de onda son bloqueados, así que el problema de optimización de la red correspondiente es minimizar la probabilidad de bloqueo [3].

Aunque las redes WRON convencionales ofrecen gran cantidad de ventajas, cuentan con la desventaja de su rigidez. Actualmente, estas redes requieren la asignación completa de una longitud de onda para establecer una conexión, incluso si el tráfico entre los nodos finales no es suficiente para rellenar la capacidad al completo, lo que causa usos ineficientes de la capacidad. La necesidad de flexibilidad y eficiencia requiere el uso de recursos más precisos. Idealmente, una red adaptativa será lo suficientemente precisa como para proporcionar elásticamente la capacidad demandada. Tecnologías como la conmutación óptica por paquetes (OPS, *Optical Packet Switching*) y conmutación óptica por ráfagas (OBS, *Optical Burst Switching*) cumplen estos requisitos, pero todavía no son suficientemente maduras [12].

A la hora de solucionar estos problemas, la comunidad científica se ha decantado por las redes WRON elásticas, redes capaces de asignar espectros a los *lightpaths* en función de los requisitos de ancho de banda de los clientes. En estas redes se pueden variar diversos parámetros que actualmente son fijos en las redes WRON convencionales, como la tasa de transmisión, el formato de modulación o la anchura de los canales [13]. No obstante, estas redes están sujetas a otros inconvenientes y cuestiones que se relatan a continuación y que están siendo objeto de investigación.

2.1. Modulación

Las redes WRON convencionales asignan los recursos espectrales a rutas ópticas sin considerar el formato de modulación adecuado, lo que lleva a usos ineficientes del espectro. Sin embargo, las redes WRON elásticas aprovechan su flexibilidad para mejorar su eficiencia espectral a través del uso de determinados formatos de modulación [13]. Existen tres formatos que sobresalen con respecto al resto: OFDM (*Orthogonal Frequency-Division Multiplexing*), N-WDM (Nyquist WDM) y OAWG (*Optical Arbitrary Waveform Generation*). En OFDM, formato de modulación más extendido, los datos se transmiten sobre múltiples subportadoras ortogonales. Recientemente, se ha investigado en una versión óptica de OFDM para solventar algunos problemas de transmisión acaecientes en la fibra óptica. Aparte de las ventajas de ofrecer una baja tasa de símbolo en cada subportadora y detección coherente, OFDM aporta una eficiencia espectral única, permitiendo que los espectros de subportadoras adyacentes se superpongan debido a su modulación ortogonal. Además, OFDM permite la transmisión de anchos de banda elásticos, que se lleva a cabo asignando un número variable de subportadoras para una transmisión [12]. Respecto a N-WDM, usa subportadoras ópticas con un espectro de frecuencias prácticamente rectangulares cercano al límite de Nyquist, obteniendo transmisiones sin interferencias intersímbolo. Por último, OAWG es capaz de crear formas de onda de datos de grandes anchos de banda en cualquier formato de modulación [6].

2.2. Problema RSA

El uso de OFDM como un formato de modulación espectralmente muy eficiente y con ancho de banda variable puede proporcionar una precisión escalable y flexible.

Sin embargo, este concepto supone nuevos retos, ya que los algoritmos RWA tradicionales ya no son aplicables. Para resolver este problema, hacen falta nuevos algoritmos de encaminamiento y asignación espectral (RSA, *Routing and Spectrum Allocation*) [12]. Si además hubiera flexibilidad en la selección del formato de modulación, podríamos hablar del problema RMLSA (*Routing, Modulation Level and Spectrum Allocation*) [6].

En el problema RSA, un grupo de *slots* contiguos se asigna a una conexión, en lugar de un grupo de longitudes de onda, como ocurre en RWA. Estos *slots* deben estar seguidos para cumplir con la restricción de contigüidad del espectro. No obstante, si no hay suficientes *slots* contiguos en la ruta deseada, la conexión se puede dividir en pequeñas demandas, donde cada una de esas demandas necesitará un menor número de *slots* [13]. A este concepto lo hemos denominado *disjoint* a lo largo del trabajo.

El problema RSA se suele fragmentar en dos subproblemas: el encaminamiento y la asignación del espectro. En el encaminamiento, se pueden utilizar las mismas técnicas que para el problema RWA. No obstante, si queremos evitar la fragmentación del espectro, hay técnicas más complejas, tal y como veremos en el siguiente apartado. En cuanto a la asignación del espectro, se han desarrollado métodos como *first-fit*, *most-used*, *least-used* o *exact-fit*. La mayoría de autores coinciden en que la mejor técnica es *exact-fit*, que consiste en buscar un hueco exacto. Existe otra forma de resolver el problema RSA, conocida como *joint* RSA, que consiste en resolverlo de forma conjunta, es decir, teniendo en cuenta el encaminamiento y la asignación del espectro a la vez. Algunos autores han tratado de resolver este problema y el problema de la modulación (RMLSA) de forma conjunta. Sin embargo, estos algoritmos requieren un mayor tiempo de computación y no son muy adecuados para la asignación dinámica de *lightpaths* [13].

2.3. Desfragmentación del espectro

La fragmentación es un aspecto intrínseco en las redes ópticas elásticas. Como las señales se establecen y se liberan de forma dinámica, aparecen inevitablemente huecos en el espectro, complicando la asignación de algunos *slots*, que quedan aislados. El problema de la fragmentación se aborda tratando de aliviarla, o llevando a cabo un proceso de desfragmentación, que resulta más efectivo. Su objetivo es reorganizar los *lightpaths* existentes para hacer hueco a aquellos que, de otra forma, serían bloqueados. El alivio de la fragmentación se puede conseguir con algoritmos RSA más sofisticados. Si, por el contrario, utilizamos un proceso de desfragmentación, éste mejora el uso del espectro tomando medidas activas en el tráfico establecido, con el fin de reducir los huecos aislados. Existen dos opciones, tal y como se muestra en la Figura 2: reencaminar las rutas ópticas con o sin reasignación del espectro, o reasignar los *slots* manteniendo la misma ruta óptica. Como esta reorganización conlleva normalmente un reencaminamiento o una reasignación del espectro de las conexiones existentes, estas técnicas requieren de mucho tiempo de computación. Por tanto, un requisito fundamental es no

interrumpir el servicio durante la fase de reconfiguración. La primera opción implica la interrupción del servicio, mientras que la segunda no. Además, ambas opciones precisan del uso de sintonización avanzada en los láseres y procesamiento digital de las señales (DSP, *Digital Signal Processing*) [6, 14].

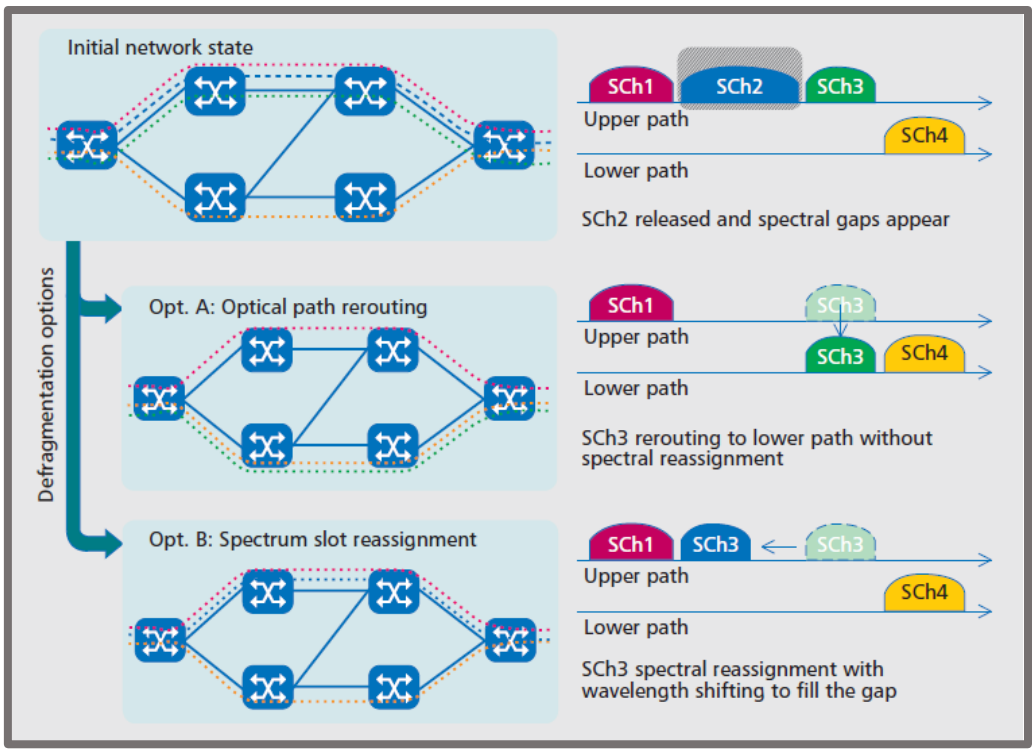


Figura 2. Dos técnicas de desfragmentación del espectro. La primera opta por reencaminar slots, manteniendo las frecuencias, mientras que la segunda reasigna su espectro en la misma ruta.

2.4. Grooming

En las redes WRON convencionales, el *grooming* se usa para multiplexar una serie de peticiones de conexión a baja velocidad en un canal a alta capacidad para mejorar la utilización del mismo. Con esta técnica se mejora el uso de recursos agregando múltiples canales eléctricos (flujos de paquetes o circuitos) en un canal óptico. Aunque las redes ópticas elásticas asignan los recursos espectrales con el ancho de banda justo para satisfacer las demandas de tráfico, el *grooming* sigue siendo esencial, entre otras cosas porque elimina el uso de bandas de guarda, ya que el tráfico se agrega en el dominio eléctrico [13].

2.5. Supervivencia

La supervivencia es un requisito esencial de las redes ópticas elásticas, principalmente por la gran cantidad de información que transmiten por unidad de segundo. Se pueden dividir en 2 técnicas: protección y restauración. La protección usa rutas de respaldo para transportar las señales ópticas tras la ocurrencia de fallo. Estas rutas se computan antes del fallo, pero se reconfiguran tras el fallo. Aunque la protección dedicada aporta mayor seguridad, la protección compartida (compartición de los recursos de protección entre conexiones con diferentes rutas

de trabajo) es más eficiente. En la restauración, las rutas de respaldo se computan dinámicamente tras el fallo, y por tanto pueden proporcionar una mayor eficiencia en términos de utilización de recursos [13].

2.6. Plano de control

Los patrones de tráfico están evolucionando, motivados principalmente por los requisitos de las redes de computación *cloud* y los centros de datos. Todo ello está llevando a los proveedores a replantearse la arquitectura de sus redes, particularmente la relación entre IP y las capas de transporte óptico. La práctica habitual de sobreabastecer la capa IP y capar la capa óptica se está volviendo obsoleta, pues están emergiendo soluciones ópticas elásticas con conmutación digital integrada. Además, es necesaria una orquestación de recursos entre diferentes dominios y fabricantes de equipos. En la capa de transporte óptica, esto requiere permitir la entrega rápida de ancho de banda de manera eficiente, tanto de recursos como económicamente, y preferiblemente con bajos tiempos de conmutación. Esto se traduce en el despliegue de soluciones con planos de control avanzados que soporten completamente el potencial de las redes ópticas elásticas. Las dos tecnologías que se han situado a la cabeza para solventar este problema son GMPLS y SDN [6].

GMPLS ofrece una representación lógica de redes con múltiples tecnologías y capas. En GMPLS existen tanto arquitecturas centralizadas como distribuidas. Las arquitecturas centralizadas son capaces de aprovechar los recursos computacionales dedicados para conmutar rutas, almacenar información sobre el estado de la red, y modificar conexiones ya activas, comportándose como un controlador de la red [6, 14].

SDN está ganando adeptos, y proporciona un plano de control mucho menos complejo que GMPLS. En SDN, un controlador centralizado está a cargo de la orquestación y abstracción efectiva de la infraestructura subyacente. Los paquetes y circuitos se ven como flujos, y los conmutadores del plano de datos se abstraen y se presentan como controladores de *software* externos ejecutando un sistema operativo de redes. Toda la lógica de control de la red está implementada en aplicaciones sobre este sistema operativo. De esta manera, no se implementan protocolos distribuidos, y la interoperabilidad entre equipos de distintos fabricantes está garantizada [6, 14].

Tanto SDN como GMPLS están convergiendo hacia una arquitectura de un controlador común para las redes ópticas. Desde el punto de vista del rendimiento, GMPLS ha demostrado que es capaz de asegurar un tiempo adecuado de recuperación ante fallos en las redes ópticas elásticas. Por otro lado, SDN permite configurar los aparatos de forma más precisa y facilita soluciones en el plano de datos para la desfragmentación de la red y la adaptación dinámica [14].

2.7. Artículos similares al objetivo del TFG

En [15], trabajan con un espectro fijo (*fixed-grid*) y sin tener en cuenta las bandas de guarda. Sus resultados se miden en función de la probabilidad de bloqueo y de la fragmentación, y comparan una serie de algoritmos de asignación de recursos, concluyendo que el más eficiente es *exact-fit*, un algoritmo que busca un hueco exacto a la capacidad demandada. Si no lo encuentra, asigna el hueco más grande existente. Ésta ha sido comparada, entre otras, con la técnica *smallest-fit*, similar a lo que nosotros hemos denominado *best-gap*, cuyas prestaciones están algo por debajo de *exact-fit*. No obstante, cabe destacar que si trabajamos con un modelo *gridless*, sería muy improbable encontrar un hueco exacto, por lo que el algoritmo *exact-fit* no sería válido.

En [16], desarrollan un algoritmo RSA estático para redes ópticas elásticas con protección dedicada. La principal diferencia respecto a este TFG es que nosotros hemos desarrollado un algoritmo RSA dinámico, también con protección dedicada.

Capítulo 3. Algoritmos implementados

El simulador de redes WRON elásticas desarrollado por el Grupo de Comunicaciones Ópticas de la Universidad de Valladolid está formado por un nodo control y una serie de nodos de la red que forman una topología arbitraria. el nodo de control es el que maneja toda la información de la red, y por él pasan todas las peticiones de establecimiento y liberación de *lightpaths*. Por tanto, se trata de una arquitectura centralizada.

En este capítulo describiremos paso a paso cada algoritmo desarrollado. El primero es el algoritmo dinámico, que ha sido adaptado para ofrecer protección a las peticiones si así se requiere. El resto son los algoritmos de asignación de recursos, que son llamados durante la ejecución del algoritmo dinámico.

3.1. Algoritmo dinámico

Lo primero que se realiza no es una parte imprescindible del algoritmo, pero sí lo es para evaluar el tiempo de cálculo necesario de cada uno de los algoritmos de asignación. De esta forma, en cuanto recibimos una petición de establecimiento de *lightpath*, se guarda el instante de tiempo en el que ha llegado dicha petición. Después, cuando el algoritmo termine, volveremos a obtener el tiempo actual y calcularemos la diferencia. Si bien esta diferencia de tiempos variará en función de las características de la máquina utilizada, todos los resultados de este TFG se han realizado con el mismo servidor para tener así una comparativa justa (máquina utilizada para el estudio: AMD Opteron 6172, 2.1 GHz, 64 GB RAM).

La siguiente tarea es extraer cierta información importante de la petición, como el nodo origen, el nodo destino y la capacidad, para después comprobar si existen transmisores disponibles en el origen y receptores en el destino. En caso negativo, se bloqueará la petición. No obstante, el objetivo de este trabajo es averiguar qué algoritmo de asignación es el más eficiente en término de consumo de espectro y no en el número de transmisores, por lo que se ha asignado un número de transmisores y receptores en cada nodo lo suficientemente alto como para que nunca se dé esta situación de bloqueo, esto es, 1000 transmisores y receptores por nodo.

A continuación, entramos en un bucle que itera en función de las rutas disponibles entre origen y destino. Puesto que este número suele oscilar en torno a la centena, hemos decidido realizar las simulaciones con un número máximo de rutas en las que comprobar si existen frecuencias. En concreto, se han hecho simulaciones con 1, 2, 3, 5 y 10 rutas máximas (K_{max}, en adelante). Con $K_{max} > 10$, los tiempos de simulación comienzan a ser muy elevados, y no se consiguen grandes ventajas.

Hagamos un pequeño inciso para explicar adecuadamente este punto. Durante la inicialización de la red, una de las tareas que se ha llevado a cabo es la de ordenar todas las rutas entre cada origen y cada destino siguiendo algún tipo de algoritmo. En el presente documento hemos seguido dos diferentes: *shortest-path* y *least-delayed-path*. En ellos se ordenan todas las rutas de menor a mayor distancia. La

diferencia está en que el primero utilizará el número de saltos como métrica mientras que en el segundo se utilizarán los retardos de propagación de los enlaces. Para ambos casos, usamos un algoritmo basado en el ordenamiento en burbuja [17].

Una vez conocemos la ruta, nos interesa conocer el estado de sus enlaces. Para ello, llamamos a una función que compruebe la TED (*Traffic Engineering Database*) y nos devuelva las frecuencias ocupadas conjuntas de los enlaces de la ruta. Este apartado se explica en profundidad más adelante.

Después, llamamos a la función que resolverá el problema RSA (*Routing and Spectrum Assignment*). Ésta nos devolverá las frecuencias a asignar en función del algoritmo de asignación elegido. En este trabajo se han desarrollado dieciséis algoritmos diferentes, ocho con la filosofía *first-fit*, es decir, se asignan las primeras frecuencias disponibles, y ocho con *best-gap*, esto es, se asignan los huecos más adaptados, o dicho de otra forma, aquellos cuya diferencia entre el ancho de banda disponible y el demandado (además de las bandas de guarda) es mínima. Este punto también se explica más adelante.

Cuando obtenemos las frecuencias a asignar, se pueden dar dos situaciones: que las frecuencias tengan suficiente ancho de banda como para albergar toda la capacidad requerida, o que no sean suficientes o incluso no haya ninguna porción del espectro disponible. En este segundo caso, en el que necesitamos más ancho de banda, realizaremos otra iteración del bucle con los mismos pasos relatados hasta ahora, pero utilizando el siguiente camino (ordenados por distancia, como anteriormente se ha comentado) siempre y cuando no excedamos K_{max}. En caso de exceder este número máximo de caminos, bloquearíamos la petición.

Si el ancho de banda asignado fuera suficiente, comprobaríamos si la protección está activada, esto es, si queremos reservar recursos adicionales en caso de fallo en los caminos de trabajo. En este trabajo se ha implementado tanto un método de protección 1+1 como 1:1, esto es, se envían los datos por ambos caminos (1+1) o si se produce un fallo en el enlace o en algún nodo, se conmuta automáticamente al camino de protección (1:1) [7]. Es decir, la única diferencia es utilizar un *splitter* o un *switch* en el origen, tal y como se muestra en la Figura 3, por lo que no influye en la programación. Si la protección no está activada, se establece el *lightpath*. En caso contrario, pasaríamos por un proceso similar al ya descrito, con una diferencia puntual. Se trata del camino de protección, que debe ser totalmente diferente al camino de trabajo, esto es, no puede coincidir ningún enlace. Por este motivo, tal y como se puede observar en el diagrama de flujo de la Figura 4, las rutas las iteraremos con una variable, *j*, y tendremos una variable aparte, *c*, para llevar cuenta de las rutas realmente válidas. Será esta variable *c* la que compararemos con K_{max} para saber si hemos superado el número de caminos máximo.

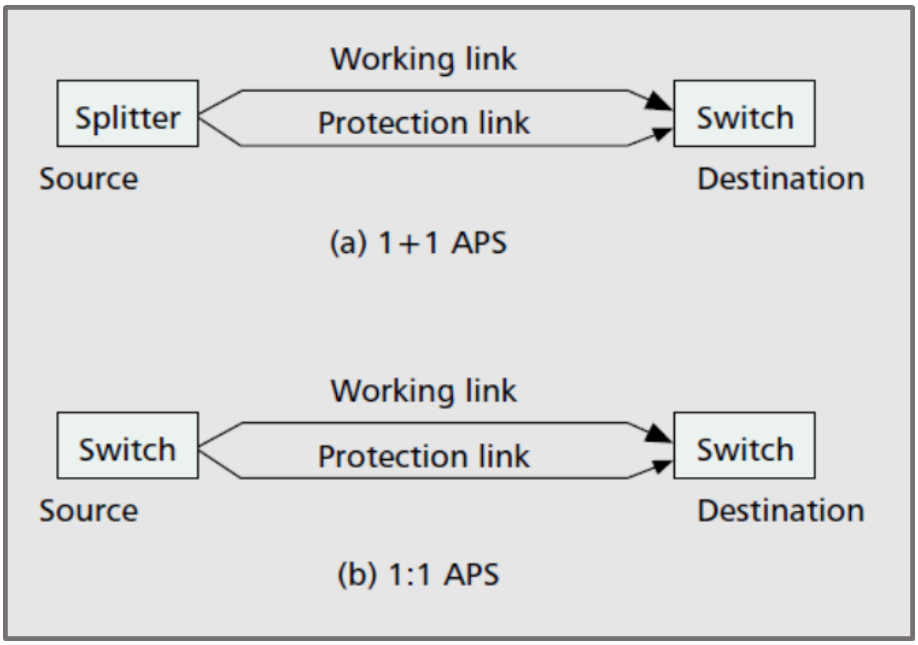


Figura 3. Arquitectura básica de protección 1+1 y 1:1.

En la Figura 4 se muestra un diagrama de flujo donde se detalla el proceso desde que llega la petición de *lightpath* hasta su establecimiento o bloqueo.

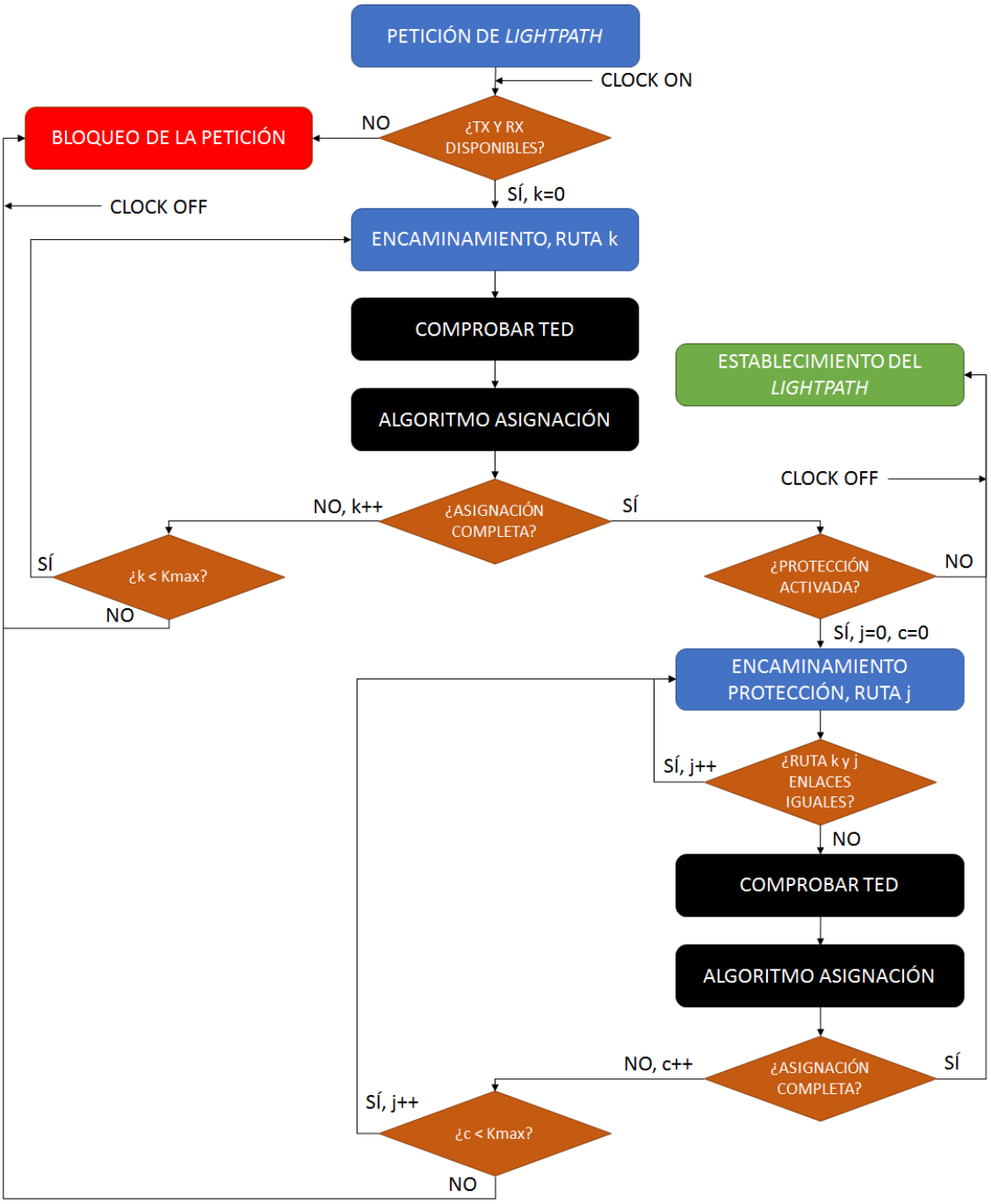


Figura 4. Diagrama de flujo del algoritmo dinámico implementado.

Comprobar TED

En este apartado vamos a explicar la caja negra "Comprobar TED" que aparece en el diagrama de flujo de la Figura 4. Esta función recibe los enlaces de la ruta, y devuelve la Matriz de Capacidad Conjunta (MCC, en adelante), esto es, las frecuencias ocupadas en uno o varios canales de todos los enlaces que conforman la ruta. Para ello, se recogen los datos de la tabla de estado de ocupación del espectro de los canales de longitud de onda en las fibras (TED, en adelante), que se encuentra en el módulo llamado "Estado". En esta función, se siguen 3 pasos:

1. Se recuperan a través de la TED los datos de ocupación de frecuencias de cada enlace de la ruta, usando un bucle que recorre cada enlace.
2. Cuando tenemos todos los datos de la ruta almacenados en una sola matriz, ordenamos dicha matriz para que las frecuencias menores se encuentren en las primeras posiciones, y las mayores en las últimas.
3. Por último, se eliminan los solapamientos entre frecuencias.

El punto 3 puede ser el más complicado de entender. Por ello, vamos a ilustrarlo a través del siguiente ejemplo. Supongamos una ruta entre dos nodos formada por los enlaces 0, 5 y 19. En el enlace 0 están ocupadas las frecuencias desde 0 GHz hasta 300 GHz, mientras que el enlace 5 tiene ocupadas las frecuencias desde 0 GHz hasta 100 GHz, y desde 800 GHz hasta 900 GHz. Por último, en el enlace 19 están ocupadas las frecuencias desde 200 GHz hasta 450 GHz, y desde 450 GHz hasta 600 GHz. En este caso, la MCC que obtendremos indicará que las frecuencias ocupadas son desde 0 GHz hasta 600 GHz, y desde 800 GHz hasta 900 GHz, tal y como se observa en la Figura 5. Esto es debido a que por todos los enlaces de un *lightpath* deben transmitirse las mismas longitudes de onda, dado que suponemos que no hay convertidores de longitud de onda, y por ello, solamente nos son útiles frecuencias que no hayan sido utilizadas previamente en ningún enlace. Cabe destacar que esta función es válida independientemente del algoritmo que estemos usando, y que posteriormente veremos.

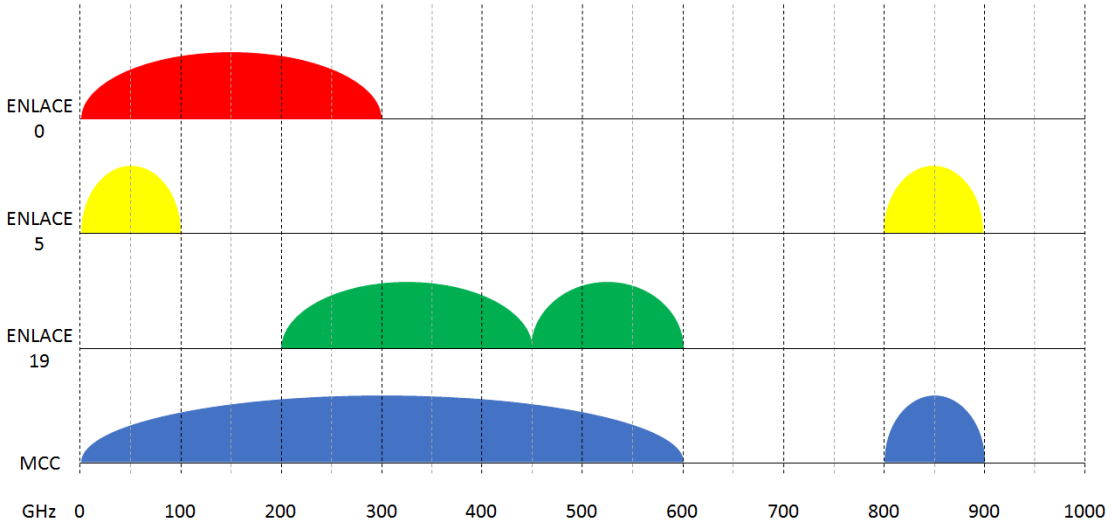


Figura 5. Ejemplo del estado de tres enlaces que conforman una ruta y MCC resultante.

3.2. Algoritmos de asignación

A continuación, vamos a explicar de manera detallada la caja negra "Algoritmo Asignación", que aparece en la Figura 4. Se han programado dieciséis algoritmos distintos en los que se varían los siguientes parámetros entre los valores que se indican:

- Ancho de banda compacto:
  - Sí – Algoritmos *Joint*
  - No – Algoritmos *Disjoint*
- Uso de rejilla de canales:
  - Sí – Algoritmos *Fixed-Grid*
  - No – Algoritmos *Gridless*
- Búsqueda del hueco más adaptado a la demanda:
  - Sí – Algoritmos *Best-Gap*
  - No – Algoritmos *First-Fit*
- Uso de rutas de protección:
  - Sí – Algoritmos *Protected*
  - No – Algoritmos *Unprotected*

De forma resumida se pueden ver en la Tabla 1:

		FIRST-FIT		BEST-GAP	
		FIXED	GRIDLESS	FIXED	GRIDLESS
UNPROTECTED	JOINT	JSF-FF	JSG-FF	JSF-BG	JSG-BG
	DISJOINT	DSF-FF	DSG-FF	DSF-BG	DSG-BG
PROTECTED	JOINT	P_JSF-FF	P_JSG-FF	P_JSF-BG	P_JSG-BG
	DISJOINT	P_DSF-FF	P_DSG-FF	P_DSF-BG	P_DSG-BG

Tabla 1. Cuadro resumen de los algoritmos implementados.

Se puede ver en la Tabla 1 que se han realizado todas las combinaciones posibles, obteniendo un total de 16 algoritmos. Procedemos a explicar qué significan los posibles valores que pueden tomar los parámetros anteriores.

Comenzamos con el ancho de banda compacto, e incluimos en la Figura 6 un ejemplo el espectro para ilustrar mejor estos conceptos.

- *Joint*: el ancho de banda asignado debe ser compacto, es decir, se asigna un único ancho de banda.
- *Disjoint*: el ancho de banda no tiene por qué ser compacto. Esto significa que podemos asignar varios anchos de banda diferentes, e incluso éstos pueden estar en diferentes rutas.

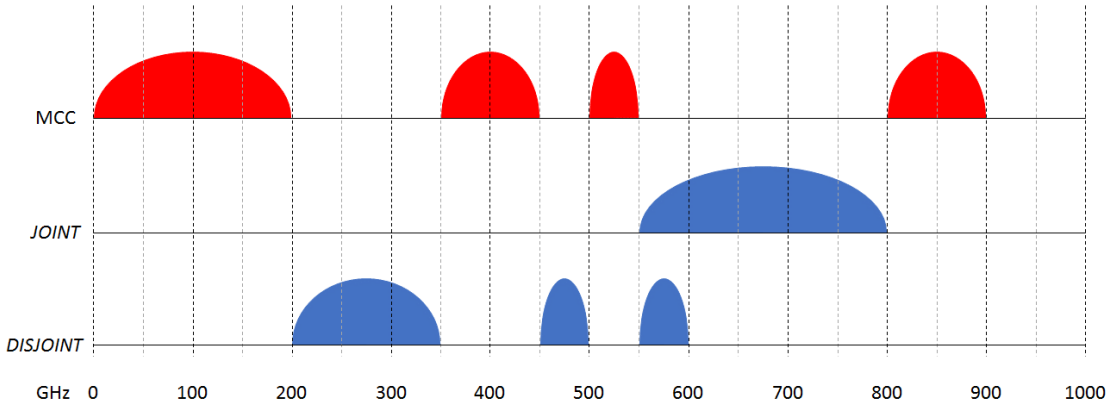


Figura 6. Ejemplo para ilustrar la diferencia entre Joint y Disjoint.

Respecto al uso de rejilla de canales, hay dos posibles valores, tal y como mostramos en el ejemplo de la Figura 7.

- *Fixed-Grid*: el espectro está ranurado, es decir, está formado por un número determinado de *slots* de igual tamaño, por lo que las asignaciones de ancho de banda se deben realizar por *slots*.
- *Gridless*: no existe ninguna restricción a la hora de asignar anchos de banda en el espectro.

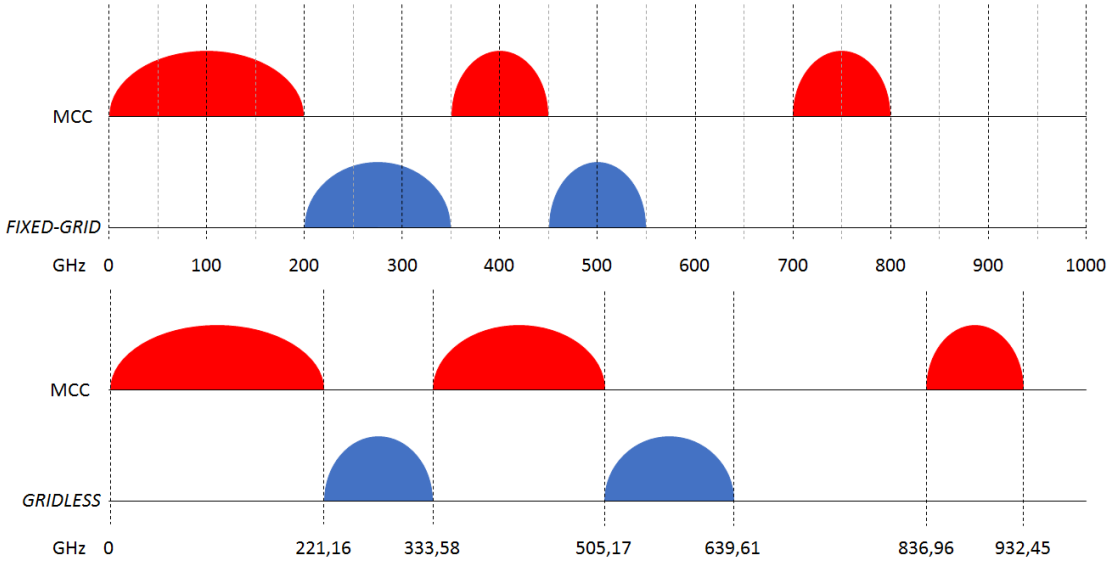


Figura 7. Ejemplo para mostrar la diferencia entre Fixed-Grid y Gridless.

En cuanto a la búsqueda del hueco más adaptado a la demanda, distinguimos entre dos posibles valores, y mostramos un ejemplo en la Figura 8.

- *Best-Gap*: entre los huecos disponibles en el espectro, asignamos aquel más adaptado a la demanda, es decir, aquel cuya diferencia entre el ancho de banda del hueco y el demandado sea mínima.
- *First-Fit*: asignamos los primeros anchos de banda disponibles en el espectro, es decir, los de menor frecuencia.

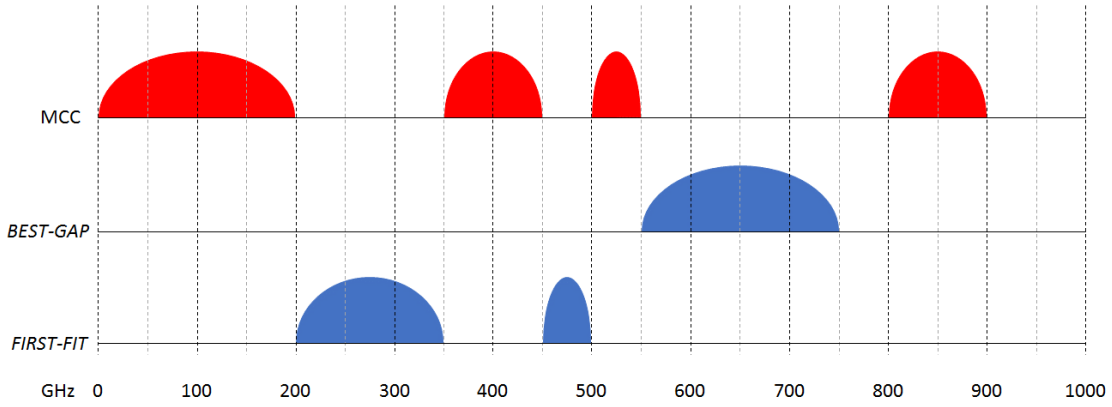


Figura 8. Ejemplo para diferenciar la metodología first-fit con best-gap.

Por último, definimos las dos posibles variantes en relación al uso de rutas de protección [8].

- *Protected*: reserva rutas adicionales que entrarán en juego en caso de fallo de algún componente de la red. Como ya hemos mencionado, en este TFG se han implementado los métodos 1+1 y 1:1.
- *Unprotected*: confía en la restauración para solventar problemas que puedan ocurrir en la red. Por tanto, una vez se ha producido un error, asigna canales que están libres para tratar de cursar todo el tráfico demandado. No obstante, este comportamiento no lo veremos en este TFG, ya que en las simulaciones no tenemos en cuenta fallos en la red.

La programación de los algoritmos, que detallamos a continuación, no variará en función de si usamos rutas de protección o no. No obstante, esto sí es relevante en el algoritmo dinámico, tal y como hemos visto en la Figura 4.

En primer lugar, explicamos los algoritmos basados en *first-fit* y después los basados en *best-gap*.

3.2.1. Algoritmos First-Fit

3.2.1.1. Joint Spectrum Fixed-Grid (First-Fit)

Se trata de un algoritmo *joint*, por lo que se asignan todos los *slots* consecutivos. Esto permite añadir una única banda de guarda, lo que implica un ahorro en el ancho de banda respecto a los algoritmos *disjoint*. En este procedimiento la asignación de frecuencias es bastante intuitiva, y la podemos dividir en varias condiciones. Que se ejecute una u otra condición dependerá de la capacidad requerida y de la Matriz de Capacidad Conjunta (MCC, en adelante), matriz que indica las frecuencias ya ocupadas en una ruta dada.

1. En primer lugar, se comprueba si la MCC está vacía, en cuyo caso se asignarán los primeros *slots* del espectro, en función de la capacidad demandada y del tamaño del *slot*.

2. Si no se cumple la condición anterior, se comprobará si los *slots* necesarios se pueden asignar entre la frecuencia mínima del espectro y el primer *slot* ocupado.
3. Si tampoco se puede llevar a cabo el paso 2, se buscaría si existe suficiente ancho de banda como para asignar los *slots* necesarios entre la frecuencia final de un *slot* ya ocupado y la frecuencia inicial del siguiente *slot* ya ocupado. Este paso se realizará con la ayuda de un bucle, para así revisar todos los huecos disponibles en el espectro que tengan *slots* ocupados tanto a su izquierda como a su derecha en el espectro.
4. Si no se diera ninguna de las anteriores condiciones, se buscaría entre la frecuencia final del último *slot* ocupado y la frecuencia máxima del espectro.

Con el cumplimiento de cualquiera de los cuatro pasos, se asignarían las frecuencias elegidas y se saldría de dicha función. Si no ocurriera esto, también se saldría de la función, pero en este caso indicando que no se han encontrado frecuencias disponibles. En la Figura 9 se observa un diagrama de flujo del algoritmo y un ejemplo del espectro en cada paso. Suponemos que los *slots* a asignar (*M*) son 2, con la capacidad requerida (flechas azules) y banda de guarda (flecha amarilla) incluidas, y que el espectro está formado por 14 *slots*, de los cuales están disponibles los que aparecen en verde y ocupados los que aparecen en naranja.

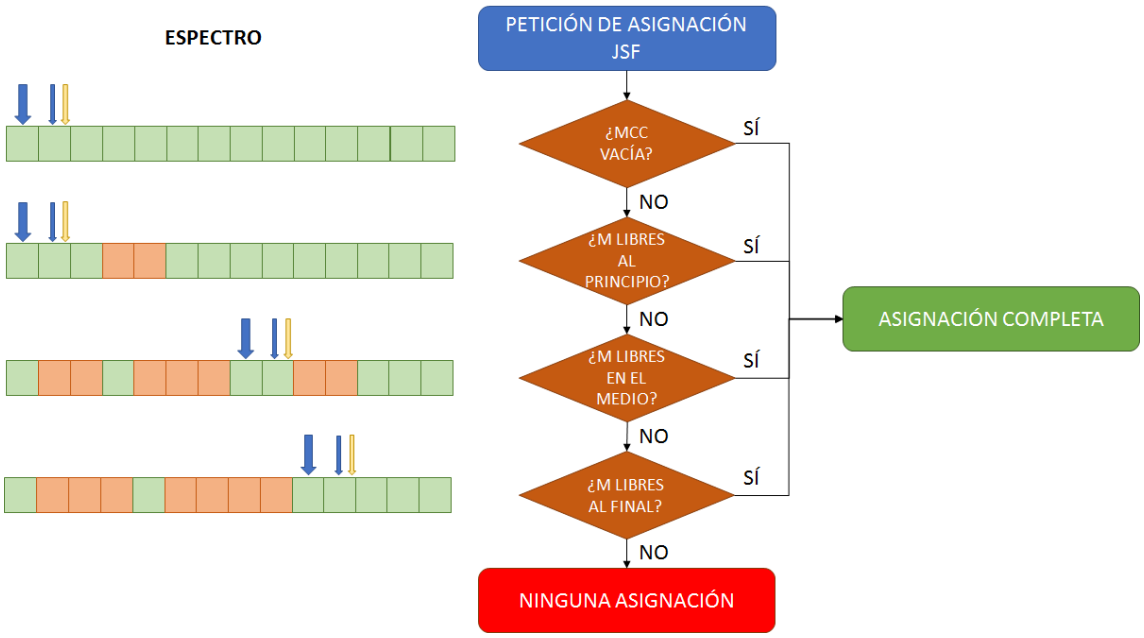


Figura 9. Diagrama de flujo de los algoritmos JSF-FF y P_JSF-FF, y ejemplo del espectro en cada paso.

3.2.1.2. Disjoint Spectrum Fixed-Grid (First-Fit)

Este algoritmo está basado, a diferencia del anterior, en una metodología *disjoint*, lo que le otorga una mayor flexibilidad a la hora de asignar frecuencias. Sin embargo, también cuenta con el inconveniente de que se usan más bandas de guarda, esto es, por cada grupo de *slots* no consecutivos se utilizará una banda de guarda. Si se diera el caso de que todos los *slots* fueran consecutivos, entonces estaríamos en el caso del algoritmo *joint*. Por otra parte, por el hecho de ser *disjoint*, se podrá asignar el ancho

de banda por varias rutas diferentes. Esto implica que se podrán utilizar rutas con enlaces coincidentes, por lo que habrá que prestar especial atención a estas rutas, para que no se asignen las mismas frecuencias en las mismas fibras. La forma de programación de este algoritmo es un tanto diferente al anterior.

1. En primer lugar, se comprueba si la MCC está vacía, en cuyo caso se asignarán los primeros *slots* del espectro, en función de la capacidad demandada y del tamaño del *slot*.
2. Si no se da la condición anterior, entramos en un bucle que itera *slot* a *slot*, es decir, en cada iteración se comprueba un *slot*, por orden de menor a mayor frecuencia. En cada una de estas iteraciones, el *slot* en cuestión cumplirá alguna de las siguientes cuatro situaciones:
    a. El *slot* se encuentra entre la frecuencia mínima del espectro y el primer *slot* reservado, en cuyo caso se asignará dicho *slot*.
    b. El *slot* en cuestión está ocupado, por lo que se saltará a la siguiente iteración del bucle.
    c. El *slot* está disponible, pero tanto a su izquierda como a su derecha en el espectro existen *slots* ocupados. En este caso, también se asignará dicho *slot*.
    d. El *slot* está situado entre la frecuencia final del último *slot* ocupado y la frecuencia máxima del espectro, por lo que también se asignará dicho *slot*.
3. En cada iteración del bucle, en caso de que se asigne un *slot,* se actualiza la capacidad que resta por asignar y se va comprobando si se ha asignado ya toda la capacidad requerida, en cuyo caso se saldrá de la función. Además, si la capacidad se hubiera fraccionado en varias rutas diferentes, cada *slot* que se va a asignar es anteriormente comprobado, para así evitar reservar dos veces el mismo *slot*.

En relación con el tercer punto, es importante destacar que, si se asignan *M slots* consecutivos, la capacidad a restar será mayor que si se asignan *M slots* no consecutivos, siendo *M* mayor que 1, ya que en el primer caso solo se utilizará una banda de guarda, mientras que en el segundo se utilizarán *M* bandas de guarda. Esto significa que, en la primera situación, donde asignamos *M slots* consecutivos, la capacidad útil será de $M * tam_{slot} - banda_{guarda}$. Sin embargo, en la segunda situación, donde asignamos *M slots* no consecutivos, la capacidad útil será $M * (tam_{slot} - banda_{guarda})$.

Una vez volvemos a la función principal, si la capacidad asignada es menor a la capacidad requerida, se busca otra ruta en la que se pueda completar la asignación de la capacidad, siempre y cuando no se supere el valor de K_{max}. Para una mejor comprensión del algoritmo, se muestra en la Figura 10 un diagrama de flujo del mismo y un ejemplo del espectro en cada paso. En el ejemplo, se puede observar que, si se asignan todos los *slots* de forma consecutiva, 4 *slots* son suficientes para albergar toda la capacidad requerida (flechas azules) y la banda de guarda (flecha amarilla). Sin embargo, si se asignan varios *slots* no consecutivos, debido al incremento de bandas de guarda, son necesarios 5 *slots.*

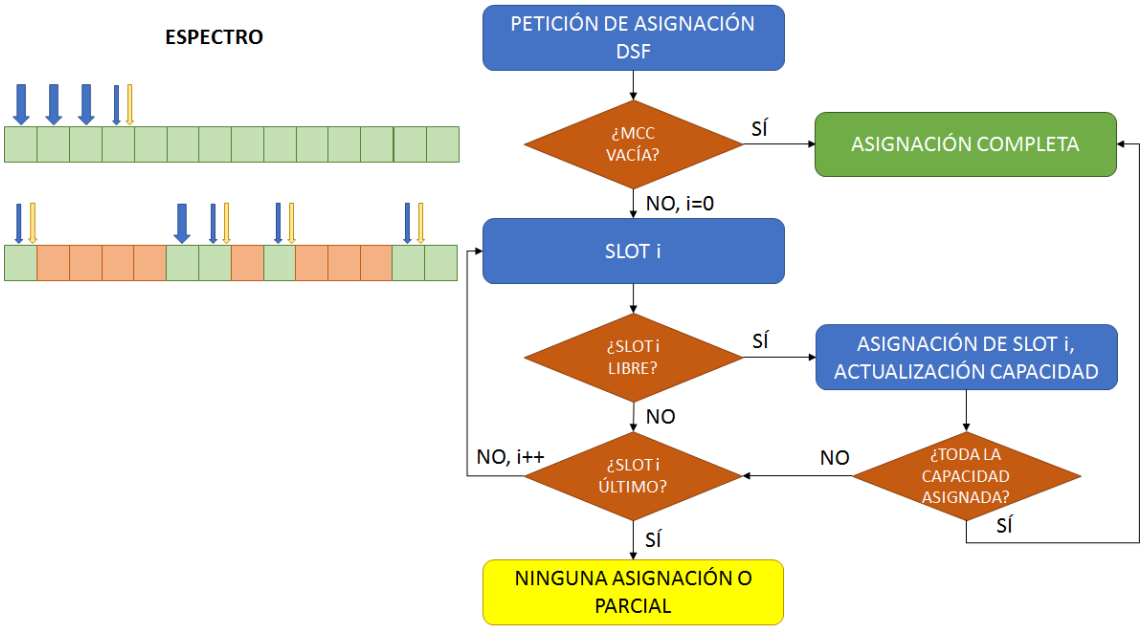


Figura 10. Diagrama de flujo de los algoritmos DSF-FF y P_DSF-FF, y ejemplo del espectro en cada paso.

3.2.1.3. Joint Spectrum Gridless (First-Fit)

En contraposición con los algoritmos anteriores, en éste, al ser *gridless*, se consigue un ahorro significativo de ancho de banda, ya que no estamos sujetos a las restricciones de las rejillas de canales. El algoritmo de programación es similar al usado en el algoritmo *JSF-FF*, y es el siguiente:

1. En primer lugar, se comprueba si la MCC está vacía, en cuyo caso se asignarán las primeras frecuencias del espectro, en función de la capacidad que requiera.
2. Si no se cumple la condición anterior, se comprobará si el ancho de banda necesario se puede asignar entre la frecuencia mínima del espectro y la frecuencia inicial del primer ancho de banda ocupado, teniendo en cuenta que entre la frecuencia final asignada y el primer ancho de banda ocupado debe haber como mínimo un hueco equivalente a una banda de guarda.
3. Si tampoco se puede llevar a cabo el paso 2, se buscaría si existe suficiente capacidad como para asignar el ancho de banda necesario entre la frecuencia final de un ancho de banda ya ocupado y la frecuencia inicial del siguiente ancho de banda ocupado. En este caso también hay que tener en cuenta que exista una banda de guarda, tanto entre el anterior ancho de banda ocupado y el principio de la asignación como entre el final de la asignación y el siguiente ancho de banda ocupado. Este paso se realizará con la ayuda de un bucle, para así revisar todos los anchos de banda disponibles que tengan frecuencias ocupadas tanto a su izquierda como a su derecha en el espectro.
4. Si no se diera ninguna de las anteriores condiciones, se buscaría entre la frecuencia final del último ancho de banda ocupado y la frecuencia máxima del espectro, también dejando un espacio equivalente a una banda de guarda

entre el último ancho de banda ocupado y la frecuencia inicial de la asignación.

Con el cumplimiento de cualquiera de los cuatro pasos, se asignarían las frecuencias elegidas y se saldría de dicha función. Si no ocurriera esto, también se saldría de la función, pero en este caso indicando que no se han encontrado frecuencias disponibles. En la Figura 11 se observa un diagrama de flujo del algoritmo y un ejemplo del espectro en cada paso. Suponemos que el ancho de banda requerido es *M* (óvalo azul), y *BG* es equivalente a una banda de guarda (óvalos amarillos). Los óvalos naranjas representan las frecuencias ya ocupadas del espectro.

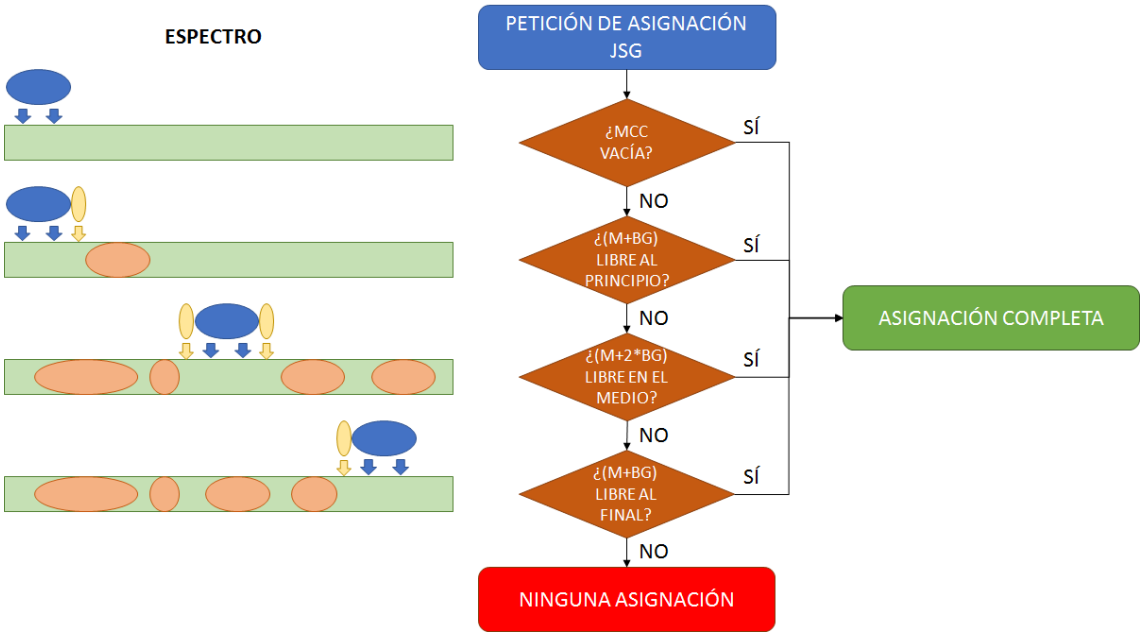


Figura 11. Diagrama de flujo de los algoritmos JSG-FF y P_JSG-FF, y ejemplo del espectro en cada paso.

3.2.1.4. Disjoint Spectrum Gridless (First-Fit)

El algoritmo de programación usado es el siguiente:

1. En primer lugar, se comprueba si la MCC está vacía, en cuyo caso se asignarán las primeras frecuencias del espectro, en función de la capacidad que requiera.
2. Si no se cumple la condición anterior, se comprobará si existe hueco entre la frecuencia inicial del espectro y la frecuencia del primer ancho de banda ocupado. Este espacio debe ser suficiente como para asignar capacidad útil, es decir, tiene que haber más hueco que el ancho de una banda de guarda. En ese caso, se pueden dar dos situaciones diferentes:
    a. Si el hueco es suficiente como para albergar la capacidad total demandada además de una banda de guarda, se asignan dichas frecuencias y concluye la función.
    b. Si el hueco no es suficiente, se asigna la fracción correspondiente, se actualiza la capacidad restante y se siguen buscando más frecuencias disponibles.

3. Si es necesario más ancho de banda (o bien no se cumple el paso 2, o bien se da el paso 2.b), se dan dos situaciones similares a las descritas en el paso 2 (situaciones a y b), con la diferencia de que en este caso se produce entre la frecuencia final de un ancho de banda ya ocupado y la inicial del siguiente ocupado. Así se van recorriendo en bucle y asignando aquellos anchos que cumplen la condición de ser mayores que dos veces la banda de guarda, puesto que se debe dejar tanto a la izquierda como a la derecha.
4. Si todavía se necesitara más ancho de banda, se buscaría de manera similar a las situaciones a y b ya descritas, solamente que entre la frecuencia final del último ancho de banda ocupado y la frecuencia máxima del espectro, por lo que el espacio deberá ser mayor a una banda de guarda.

Si la capacidad se hubiera fraccionado, cada ancho de banda que se va a asignar es anteriormente comprobado, para evitar reservar dos veces las mismas frecuencias. Una vez volvemos a la función principal, si la capacidad asignada es igual a la capacidad requerida, se completa la asignación y se termina el algoritmo. Si, por el contrario, no se ha conseguido asignar toda la capacidad, se busca otra ruta en la que se pueda completar dicha asignación, siempre y cuando no se supere K_{max}. Se muestra en la Figura 12 un diagrama de flujo del algoritmo y un ejemplo del espectro en cada paso.

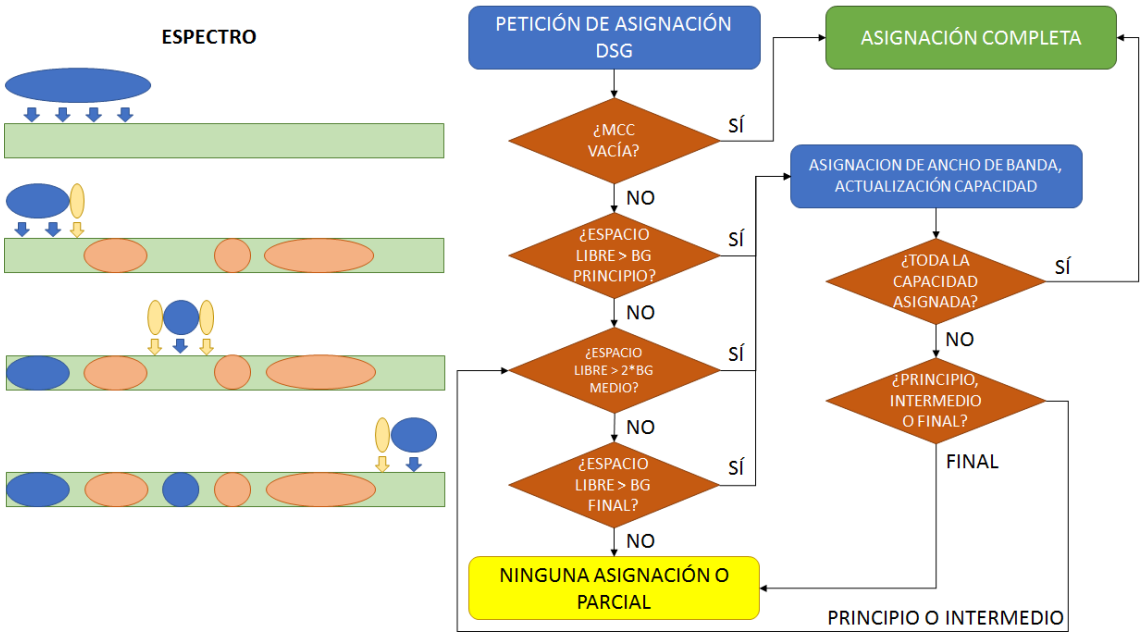


Figura 12. Diagrama de flujo de los algoritmos DSG-FF y P_DSG-FF, y ejemplo del espectro en cada paso.

3.2.2. Algoritmos Best-Gap

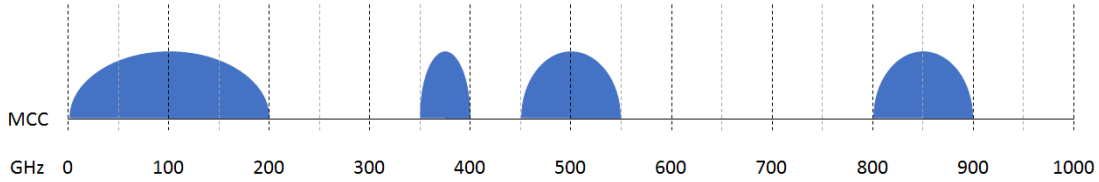
Antes de comenzar con los algoritmos relacionados con *best-gap*, se va a explicar una función que se utiliza en todos ellos, tal y como se verá más adelante.

Ordenar huecos

En este apartado vamos a explicar la caja negra "Ordenar Huecos" que aparece en todos los diagramas de flujo de los algoritmos con *best-gap*. Esta función recibe la Matriz de Capacidad Conjunta (MCC) y devuelve la Matriz de Huecos Libres (MHL, en adelante). En la primera columna de la MHL se almacenan ordenados de mayor a menor los anchos de banda libres entre todos los enlaces que conforman la ruta, mientras que en la segunda columna aparece su índice respecto a la MCC. En esta función se siguen 3 pasos:

1. Se crea una nueva matriz, *huecos_libres*, de dos columnas y una fila más que la MCC. Si la MCC estuviera vacía, entonces se devolvería la matriz *huecos_libres* de tamaño nulo.
2. Se van almacenando los anchos de banda libres y sus índices, es decir, se guarda en la primera fila con el índice 0 la diferencia entre la primera frecuencia inicial de la MCC y 0; en la segunda fila, se guarda con el índice 1 la diferencia entre la segunda frecuencia inicial de la MCC (si no existe, se utiliza la frecuencia máxima) y la primera frecuencia final, y así sucesivamente.
3. Se ordenan las filas de mayor a menor ancho de banda disponible a través del algoritmo de ordenamiento en burbuja.

Se ilustra el funcionamiento de esta caja negra a través del siguiente ejemplo. Supongamos que el argumento de entrada de la función es la MCC que aparece en la Figura 13. Puesto que dicha MCC cuenta con 4 anchos de banda ocupados (cuatro filas), la MHL contará con 5 filas y, por tanto, con índices del 0 al 4. El índice 0 tendrá un hueco libre de 0 GHz, puesto que es la diferencia entre la primera frecuencia inicial ocupada (0 GHz) y 0 GHz. El índice 1 será aquel hueco que se encuentra entre 200 GHz y 350 GHz, esto es, 150 GHz. Así seguimos hasta el último, el índice 4, que será de 100 GHz, pues resulta de la diferencia entre la frecuencia máxima del espectro (1000 GHz) y la última frecuencia final ocupada (900 GHz). Por último, se ordenan las filas de mayor a menor hueco libre. Como resultado, obtendremos la tabla que se muestra en la Figura 13.



Huecos_libres[i][j]	j = 0	j = 1
i = 0	250 (800 – 550)	3
i = 1	150 (350 – 200)	1
i = 2	100 (1000 – 900)	4
i = 3	50 (450 – 400)	2
i = 4	0 (0 – 0)	0

Figura 13. Ejemplo de una MCC, y MHL resultante.

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

Este algoritmo es diferente con respecto al algoritmo JSF-FF, ya que en este caso la filosofía de asignación es *best-gap.* No obstante, el espectro sigue siendo igual, en el sentido de *joint* y *fixed-grid*, por lo que las explicaciones y conclusiones en cuanto a estos términos son igualmente válidas aquí. El procedimiento programado es el siguiente:

1. En primer lugar, se llama a la función "Ordenar Huecos", que devuelve la MHL.
2. A continuación, se comprueba si la MHL está vacía, en cuyo caso se asignarán los primeros *slots* del espectro, en función de la capacidad demandada y del tamaño del *slot*.
3. Si no se cumple la condición anterior, entramos en un bucle que itera hueco a hueco, es decir, en cada iteración se comprueba un hueco de la MHL, que están ordenados de mayor a menor ancho de banda. En cada una de estas iteraciones se pueden dar dos situaciones:
    a. El hueco libre es mayor o igual que el ancho de banda necesario (banda de guarda incluida) para albergar la capacidad requerida: en este caso, se comprobará si es el último hueco de la MHL. Si lo es, se asignarán dichas frecuencias (sabemos cuáles son gracias al índice). Si no lo es, saltaremos a la siguiente iteración del bucle para comprobar si hay un hueco más adaptado a la capacidad requerida.
    b. El hueco libre es menor que el ancho de banda necesario: como están ordenados de mayor a menor, si se trata del primer hueco de la MHL, significa que no hay ningún hueco mayor, así que no se producirá ninguna asignación. Si, por el contrario, no es el primer hueco, se asignará el índice del anterior hueco, que era mayor que el ancho de banda necesario y, por tanto, el más adaptado.

En la Figura 14 se observa un diagrama de flujo del algoritmo y un ejemplo del espectro en cada paso. Suponemos que los *slots* a asignar (*M*) son 2, con la capacidad requerida (flechas azules) y una banda de guarda (flecha amarilla) incluidas, y que el espectro está formado por 14 *slots*, de los cuales están disponibles los que aparecen en verde y ocupados los que aparecen en naranja.

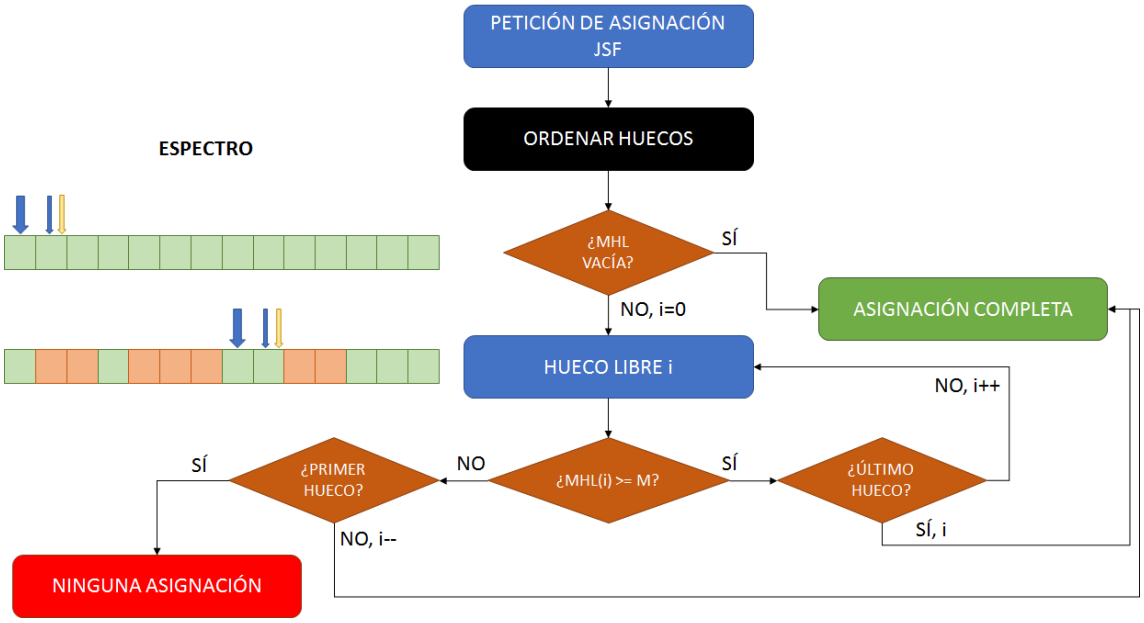


Figura 14. Diagrama de flujo de los algoritmos JSF-BG y P_JSF-BG, y ejemplo del espectro en cada paso.

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

La forma de programación de este algoritmo es similar a la del anterior, teniendo en cuenta que en este caso se pueden asignar *slots* no consecutivos.

1. En primer lugar, se llama a la función "Ordenar Huecos", que devuelve la MHL.
2. A continuación, se comprueba si la MHL está vacía, en cuyo caso se asignarán los primeros *slots* del espectro, en función de la capacidad demandada y del tamaño del *slot*.
3. Si no se cumple la condición anterior, inicializamos la variable *peq_fr* con el valor *false* y entramos en un bucle que itera hueco a hueco. En cada iteración del bucle se pueden dar dos situaciones:
    a. El hueco libre es mayor o igual que el ancho de banda necesario (banda de guarda incluida) para albergar la capacidad requerida: en este caso, se comprobará si es el último hueco de la MHL. Si lo es, se asignarán dichas frecuencias (sabemos cuáles son gracias al índice). Si no lo es, saltaremos a la siguiente iteración del bucle para comprobar si hay un hueco más adaptado a la capacidad requerida. Además, asignaremos el valor *false* a *peq_fr*, por un motivo que indicaremos más adelante.
    b. El hueco libre es menor que el ancho de banda necesario: vamos a distinguir entre dos situaciones distintas:
        i. No es el primer hueco de la MHL y *peq_fr* es falso: los huecos libres están ordenados de mayor a menor, por lo que, si se da esta circunstancia, significa que el anterior hueco era el más adaptado (situación similar al algoritmo JSF-FF explicado anteriormente) y, por tanto, se asigna todo el ancho de banda requerido al completo. Cabe destacar que, en esta condición se

puede haber entrado si el primer hueco era más pequeño que el ancho de banda requerido, y por ello es necesario haber cambiado *peq_fr* de *true* a *false*, tal y como se indica en la situación "a". Además, en este caso, ya se habrán hecho asignaciones en otros huecos, y con esta asignación se asigna toda la capacidad demandada.

ii. Es el primer hueco (activamos *peq_fr*), o *peq_fr* es verdadero: se asignará ese hueco si y solo si es mayor o igual que el tamaño de slot. Si no fuera así, saldríamos de la función puesto que ya no habrá huecos más grandes. Si asignamos, después hay que comprobar si es el último hueco de los posibles, en cuyo caso saldremos de la función habiendo realizado una asignación parcial. Si no es el último hueco, saltaremos a la siguiente iteración del bucle.

Cabe resaltar que, en cada iteración del bucle, en caso de que se asigne un *slot,* se actualiza la capacidad que resta por asignar. Por otra parte, si la capacidad se hubiera fraccionado en varias rutas diferentes, cada hueco que se va a asignar es anteriormente comprobado, para evitar reservar dos veces las mismas frecuencias (se trata de un algoritmo *disjoint*, por lo que podría darse el caso de que hubiese rutas coincidentes). Una vez volvemos a la función principal, si no se ha realizado ninguna asignación o ésta ha sido parcial, es decir, la capacidad asignada es menor a la capacidad requerida, se busca otra ruta en la que se pueda completar la asignación de la capacidad, siempre y cuando no excedamos K_{max}. Para una mejor comprensión del algoritmo, se presenta en la Figura 15 un diagrama de flujo del mismo y un ejemplo del espectro en cada paso. En el ejemplo se puede observar que, si se asignan todos los *slots* de forma consecutiva, 4 *slots* son suficientes para albergar toda la capacidad requerida (flechas azules) y la banda de guarda (flecha amarilla). Sin embargo, si se asignan varios *slots* no consecutivos, debido al incremento de bandas de guarda, son necesarios 5 *slots.*

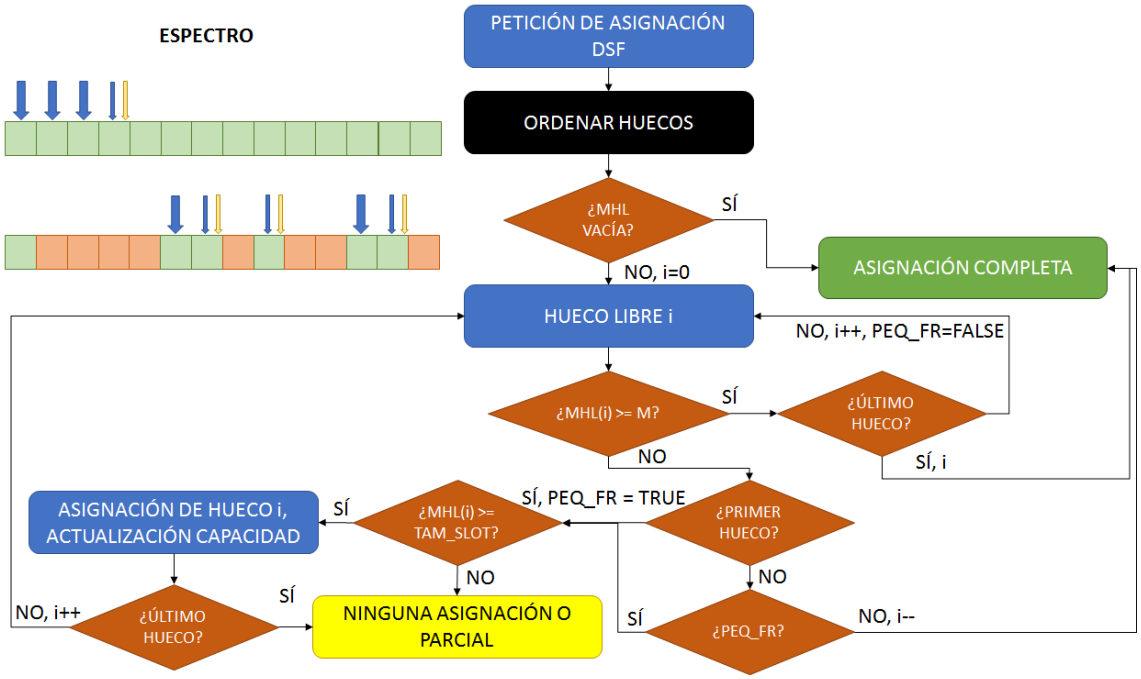


Figura 15. Diagrama de flujo de los algoritmos DSF-BG y P_DSF-BG, y ejemplo del espectro en cada paso.

3.2.2.3. Joint Spectrum Gridless (Best-Gap)

El método de programación es prácticamente idéntico al usado en el algoritmo JSF-BG. La única diferencia, que se puede apreciar en el diagrama de flujo de la Figura 16, reside en el hecho de que en este caso no tenemos un espectro ranurado. Es por ello que, a la hora de comparar el hueco disponible con la capacidad demandada, ya no tenemos *slots*, y tenemos que otorgar una capacidad igual a la disponible más una banda de guarda si el hueco está al principio o al final del espectro, y otra banda de guarda más (una a cada lado) si el hueco libre estuviera situado en otra posición. El resto del procedimiento es idéntico. En la Figura 16, además del diagrama de flujo del algoritmo, incluimos un ejemplo del espectro en cada paso. Suponemos que el ancho de banda requerido es *M* (óvalo azul), y *BG* es equivalente a una banda de guarda (óvalos amarillos). Los óvalos naranjas representan las frecuencias ya ocupadas del espectro.

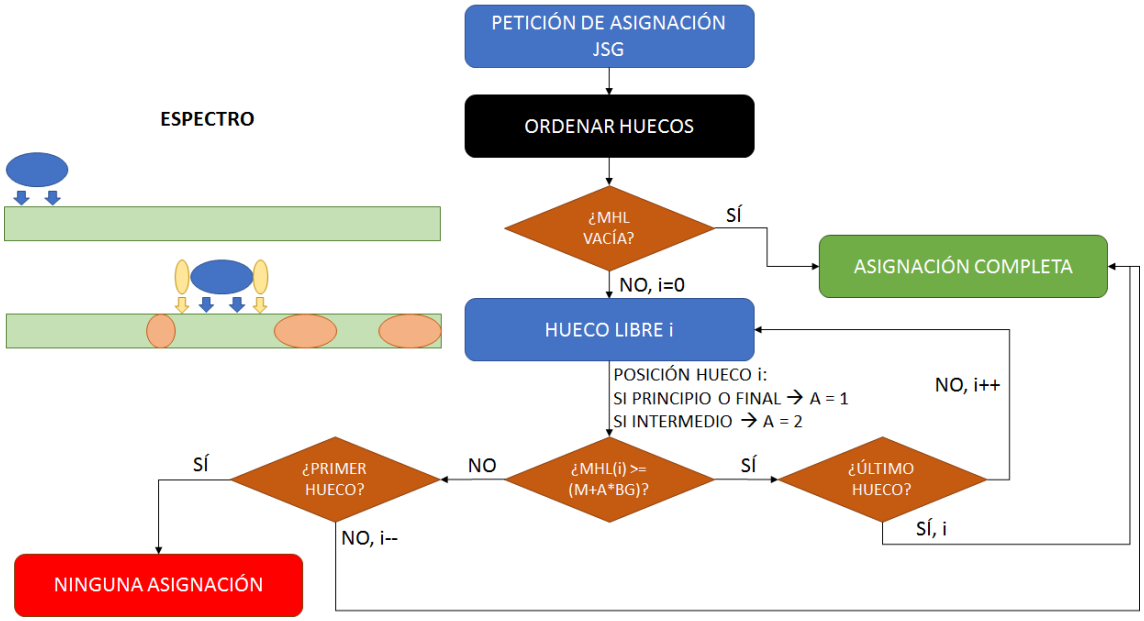


Figura 16. Diagrama de flujo de los algoritmos JSG-BG y P_JSG-BG, y ejemplo del espectro en cada paso.

3.2.2.4. Disjoint Spectrum Gridless (Best-Gap)

En este caso, el método de programación es prácticamente idéntico al usado en el algoritmo DSF-BG. La única diferencia, que se puede apreciar en el diagrama de flujo de la Figura 17, es la misma que en el algoritmo anterior (JSG-BG), y reside en el hecho de que no tenemos un espectro ranurado. El resto del procedimiento es idéntico. Además del diagrama de flujo, incluimos en la Figura 17 un ejemplo del espectro en cada paso.

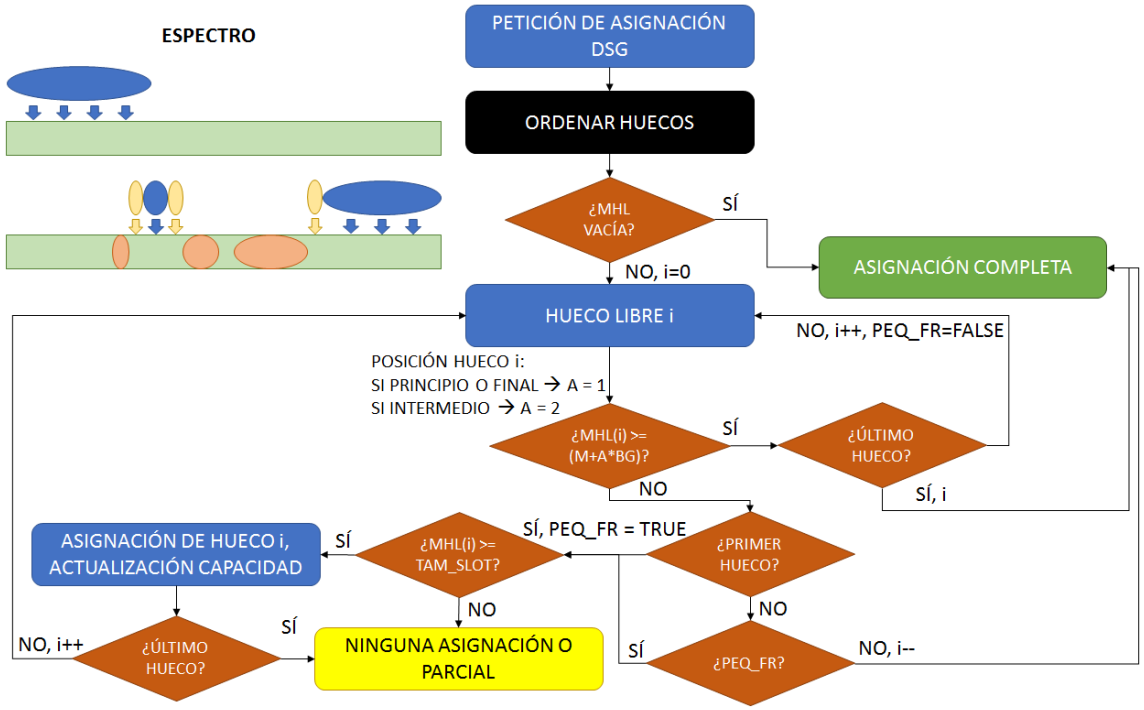


Figura 17. Diagrama de flujo de los algoritmos DSG-BG y P_DSG-BG, y ejemplo del espectro en cada paso.

Capítulo 4. Resultados

Se han realizado simulaciones utilizando un simulador desarrollado en la plataforma OMNeT++ para averiguar qué algoritmo es más eficiente y bajo qué circunstancias. Para ello, se han obtenido diversos datos, como la probabilidad de bloqueo, el tiempo medio de computación o el espectro útil medio. Los algoritmos analizados son los que se encuentran en la Tabla 1. Para cada uno de estos algoritmos se han llevado a cabo simulaciones variando una serie de parámetros:

- Orden de las rutas: durante la fase de inicialización de la red, entre otras cosas, se ordenan todas las rutas entre cada origen y destino.
    - *Shortest-Path*: se ordenan todas las rutas, siendo las primeras aquellas con el menor número de saltos desde el origen al destino.
    - *Least-Delayed-Path*: se ordenan en función del retardo que introduce cada ruta, siendo las primeras las que proporcionan un menor retardo de propagación.
- Tamaño del *slot*: únicamente para los algoritmos *Fixed-Grid*, se varía el tamaño del *slot* a algunos aceptados por [18], entre los que se encuentran 12.5 GHz, 25 GHz, 50 GHz, 100 GHz y 200 GHz.
- Número de caminos: se permitirá buscar frecuencias disponibles en 1, 2, 3, 5 o 10 rutas diferentes.
- Carga: variaremos la carga media de la red, desde 0.1 hasta 0.9, con intervalos de 0.1.

La red usada en la simulación tiene las siguientes características:

- Nombre: NSFNet
- Número de nodos: 14
- Número de transmisores y receptores: 1000

Las características comunes usadas en las simulaciones son las siguientes:

- Mínimo Ancho de Banda: 1 GHz
- Máximo Ancho de Banda: 300 GHz
- Ancho de la Banda de Guarda: 10 GHz
- Frecuencia Máxima del Espectro: 7000 GHz

Ahora pasamos a comparar cada algoritmo por separado, para después hacer una comparativa global:

4.1. Joint Spectrum Fixed-Grid (First-Fit)

Al tratarse de un algoritmo *Fixed-Grid*, en primer lugar, vamos a comprobar qué tamaño de *slot* es el más eficiente. Para ello, vamos a realizar una simulación con la protección activada (por tanto, P_JSF-FF), número de caminos máximo igual a 3, rutas ordenadas según *shortest-path* y barriendo la carga, obteniendo la

probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 18 y Figura 19, respectivamente.

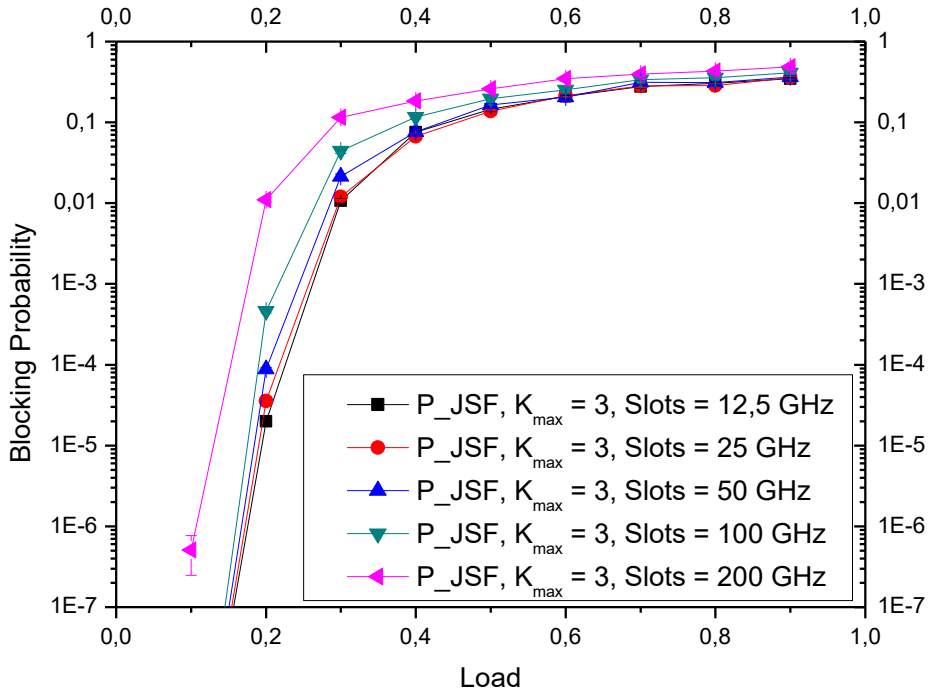


Figura 18. Probabilidad de bloqueo del algoritmo P_JSF-FF para varios tamaños de slot, con K_{max} = 3, ordenación de rutas según shortest-path y barriendo la carga.

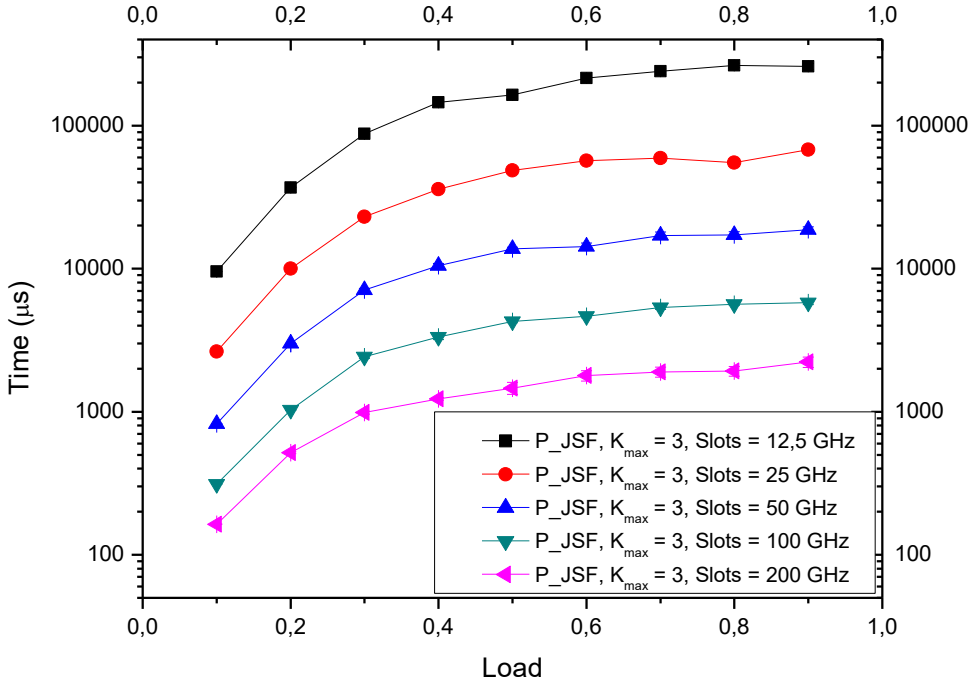


Figura 19. Tiempo de computación del algoritmo P_JSF-FF para varios tamaños de slot, con K_{max} = 3, ordenación de rutas según shortest-path y barriendo la carga.

En la Figura 18 se puede apreciar que se consigue la menor probabilidad de bloqueo en la región de trabajo (probabilidades de bloqueo inferiores a 10^{-3}) con un tamaño de *slot* de 12.5 GHz, aunque en la Figura 19 se observa que el tiempo de computación menor se produce con un tamaño de *slot* de 200 GHz, mientras que con tamaño 12.5 GHz se consigue un tiempo dos órdenes de magnitud superior. A pesar de ello, nos quedaremos con el tamaño de *slot* que minimiza la probabilidad de bloqueo, esto es, 12.5 GHz.

Cabe destacar que ambos resultados son los esperados. En relación con el tiempo, cuánto más pequeño es el tamaño del *slot*, más iteraciones tienen que recorrer los bucles, tal y como se indica en el propio método de programación ya explicado en apartados anteriores. En concreto, con un tamaño de *slot* de 12.5 GHz, se llevarán a cabo aproximadamente 16 veces más iteraciones que en el caso de 200 GHz. Acerca de la probabilidad de bloqueo, este algoritmo introduce una única banda de guarda al final de cada capacidad asignada. Por ello, el tamaño de slot de 12.5 GHz es el que menos ancho de banda desperdicia, ya que aproxima mejor las capacidades. Veamos el siguiente ejemplo para ilustrar esta afirmación. Suponiendo que queremos asignar una capacidad de 250 GHz y las bandas de guarda son de 10 GHz:

- Para un tamaño de *slot* de 12.5 GHz, necesitaremos $ceil\left(\frac{250+10}{12.5}\right) = 21\ slots$, desaprovechando, por tanto, $21 * 12.5 - 250 = 12.5\ GHz$ (incluido el ancho de la banda de guarda).
- Para un tamaño de *slot* de 25 GHz, necesitaremos $ceil\left(\frac{250+10}{25}\right) = 11\ slots$, desaprovechando $11 * 25 - 250 = 25\ GHz$.
- Para 50 GHz, necesitaremos 6 $slots$, desaprovechando 50 GHz.
- Para 100 GHz, necesitaremos 3 $slots$, desaprovechando 50 GHz.
- Para 200 GHz, necesitaremos 2 $slots$, desaprovechando 150 GHz.

Una vez visto cuál es el tamaño de *slot* más eficiente, será el que usemos de aquí en adelante, ya que este comportamiento se repetirá en todos los casos de estudio para este algoritmo.

Procedemos ahora a visualizar la influencia del número de caminos máximo. Para ello, vamos a realizar una simulación con la protección activada, variando el número de caminos, rutas ordenadas según *shortest-path*, con tamaño de *slot* de 12.5 GHz, según hemos indicado anteriormente, y barriendo la carga, obteniendo la probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 20 y Figura 21, respectivamente.

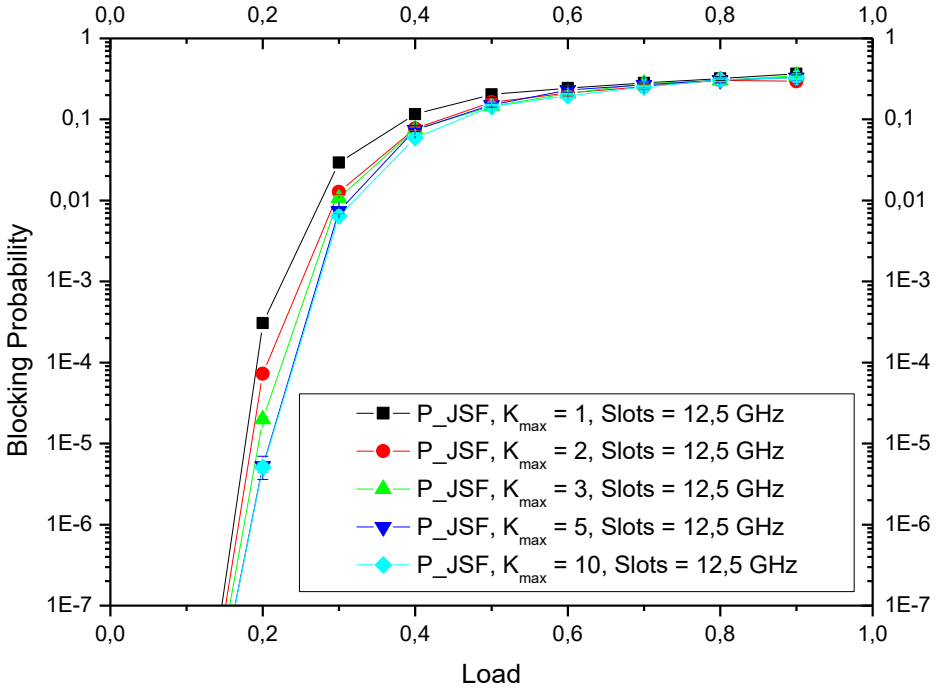


Figura 20. Probabilidad de bloqueo del algoritmo P_JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path y barriendo la carga.

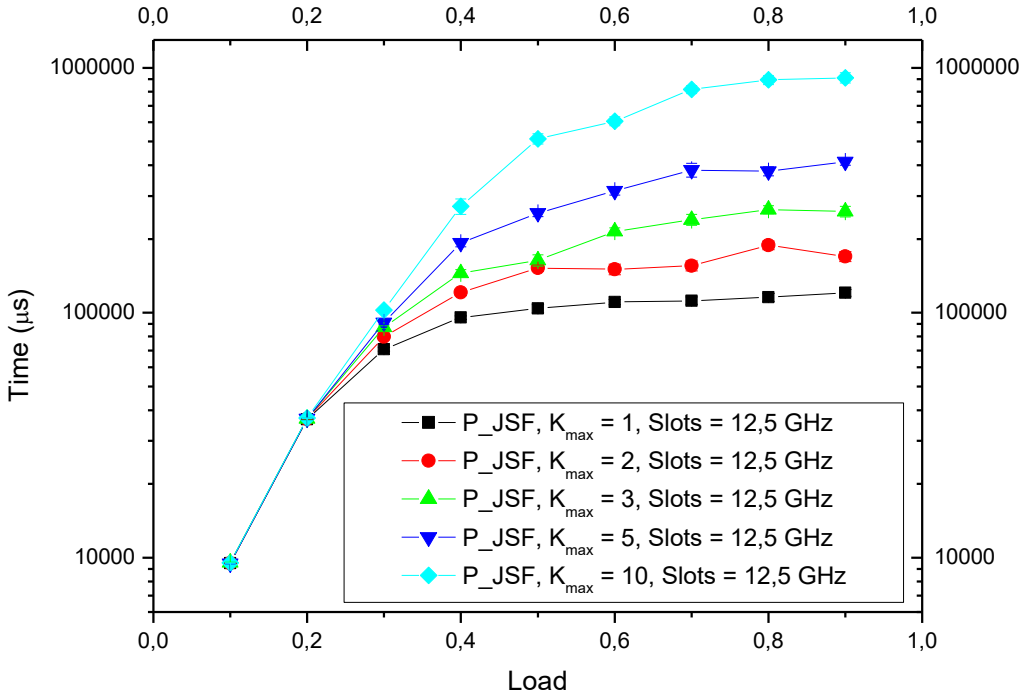


Figura 21. Tiempo de computación del algoritmo P_JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path y barriendo la carga.

Se puede observar en la Figura 20 que cuantos más caminos podamos usar, menor será la probabilidad de bloqueo, aunque, en este caso, a partir de 5 caminos la diferencia no es muy significativa. Respecto al tiempo, en la Figura 21 vemos que cuantos más caminos, mayor tiempo de computación. Estos resultados ya los habíamos anticipado, y eran de esperar. Para observar más en profundidad esta conclusión, mostramos en la Figura 22 y Figura 23 la probabilidad de bloqueo y el tiempo, respectivamente, para una carga fija de 0.3 y barriendo en este caso el número máximo de caminos.

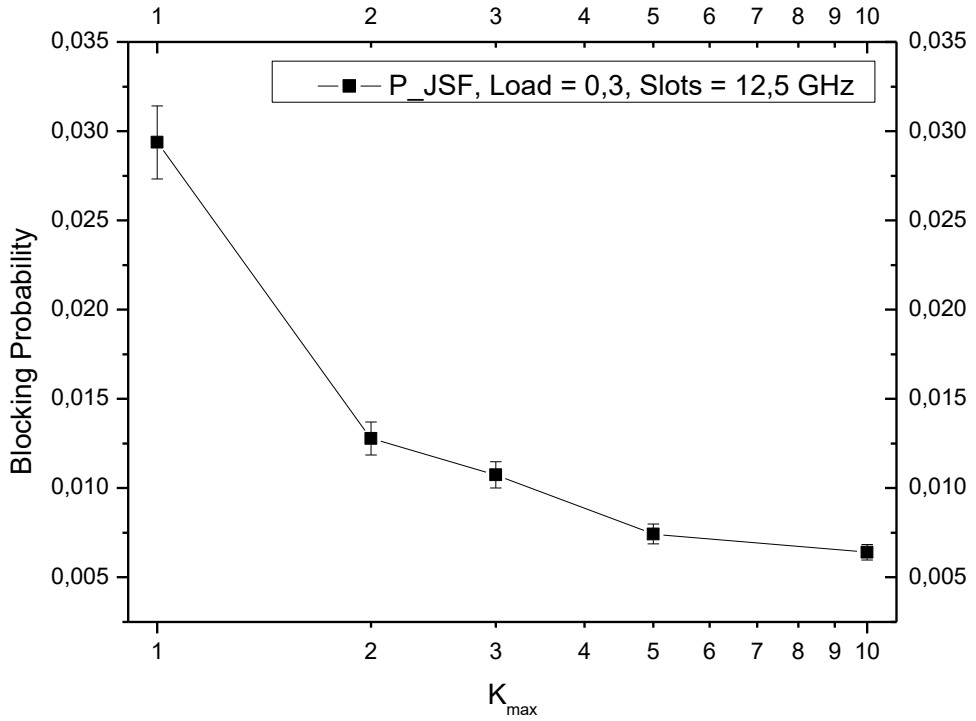


Figura 22. Probabilidad de bloqueo del algoritmo P_JSF-FF, con tamaño de slot de 12.5 GHz, carga de 0.3, ordenación de rutas según shortest-path y barriendo K_{max}.

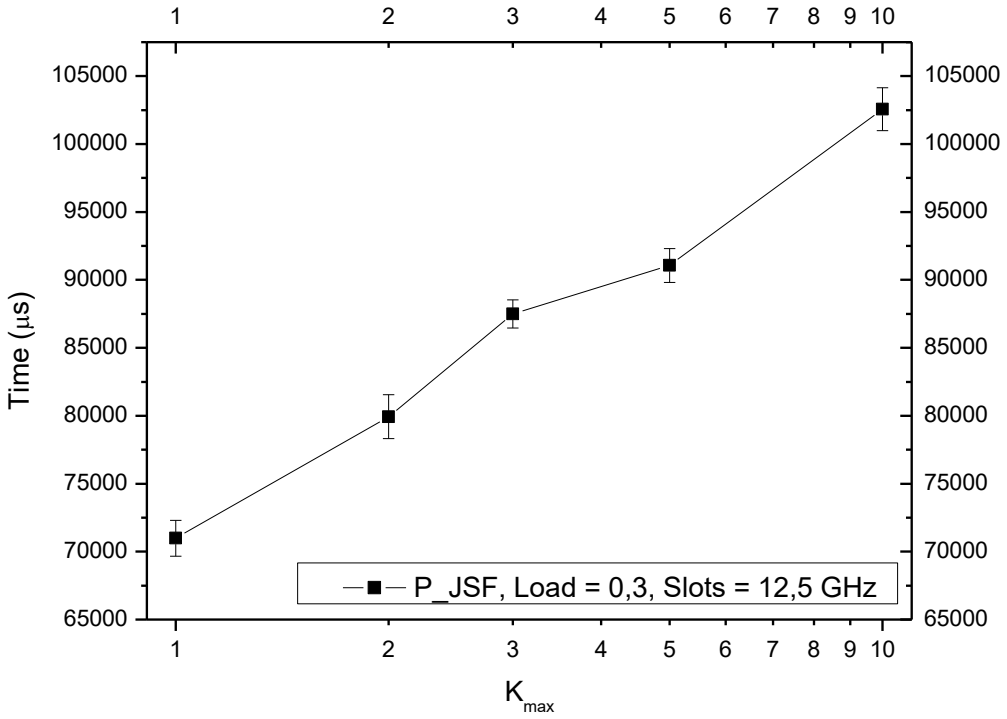


Figura 23. Tiempo de computación del algoritmo P_JSF-FF, con tamaño de slot de 12.5 GHz, carga de 0.3, ordenación de rutas según shortest-path y barriendo K_{max}.

En la Figura 22 se puede ver que el decrecimiento es aproximadamente exponencial y, por tanto, la probabilidad de bloqueo con 10 caminos será muy similar a aquella con 50. Por otra parte, en la Figura 23 observamos que el crecimiento es más o menos lineal, por lo que nos interesará un número de caminos máximo lo menor posible, para no contar con un tiempo de computación exageradamente elevado. Por tanto, podemos llegar a la conclusión que el número de caminos óptimo oscilará entre 3 y 5, ya que la probabilidad de bloqueo no mostrará cambios significativos si aumentamos los caminos máximos, y el tiempo de computación no será excesivamente grande. Esta conclusión es de suma importancia porque el establecimiento de *lightpaths* se realiza de manera dinámica, por lo que un tiempo de establecimiento demasiado elevado tiene un gran impacto en el rendimiento de la red [19].

A continuación, vamos a realizar pruebas con el algoritmo de ordenación de rutas *least-delayed-path*. Para ello, con la protección activada y con un tamaño de *slot* de 12.5 GHz, barremos la carga y obtenemos la probabilidad de bloqueo y el tiempo, según se puede ver en la Figura 24 y Figura 25, respectivamente.

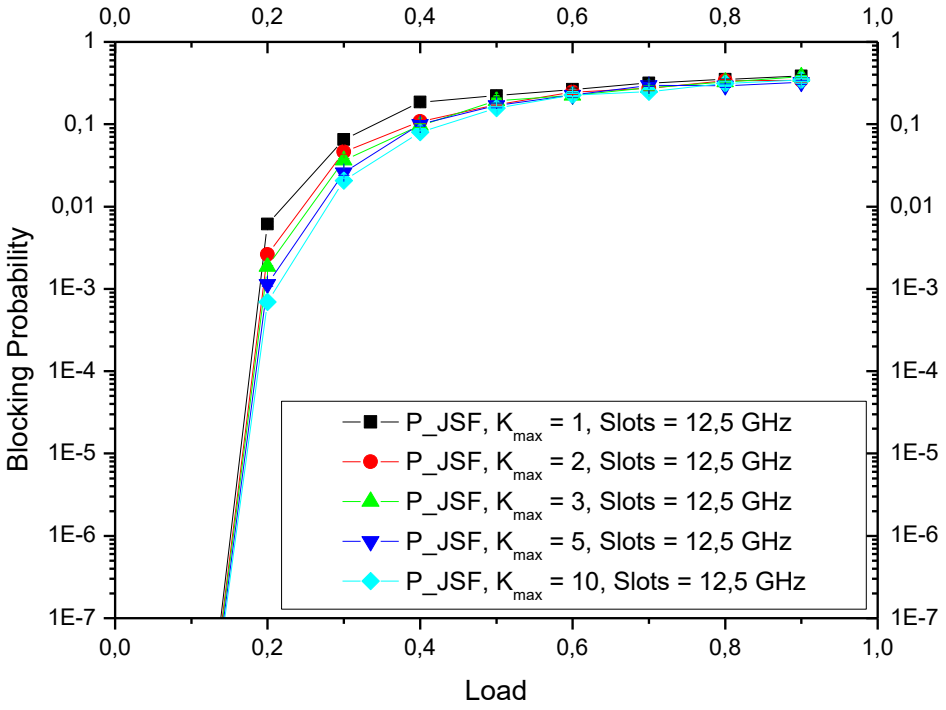


Figura 24. Probabilidad de bloqueo del algoritmo P_JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según least-delayed-path y barriendo la carga.

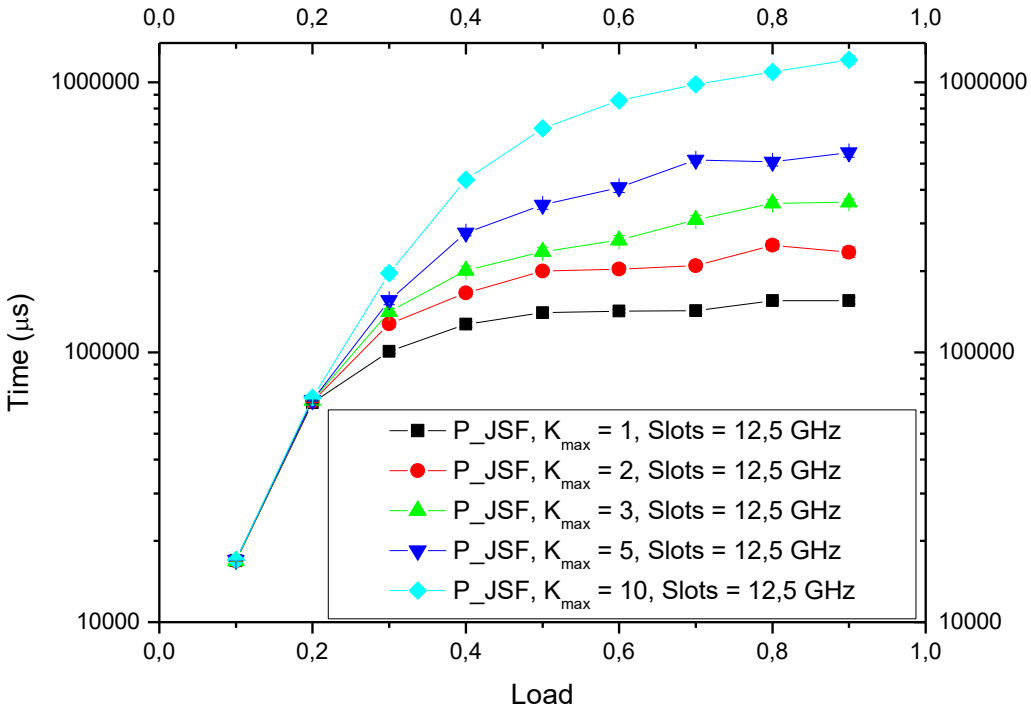


Figura 25. Tiempo de computación del algoritmo P_JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según least-delayed-path y barriendo la carga.

En este caso, vemos que obtenemos unas gráficas muy parecidas a las de la Figura 20 y Figura 21, mejorando la probabilidad de bloqueo a la vez que aumenta el número de caminos, pero también aumentando el tiempo de computación. No obstante, la gráfica realmente interesante es la que nos compara ambos algoritmos, *shortest-path* y *least-delayed-path*. Para ello, en la Figura 26 y Figura 27 mostramos las gráficas para la probabilidad de bloqueo y el tiempo con la protección activada, un número máximo de caminos de 3 y barriendo la carga.

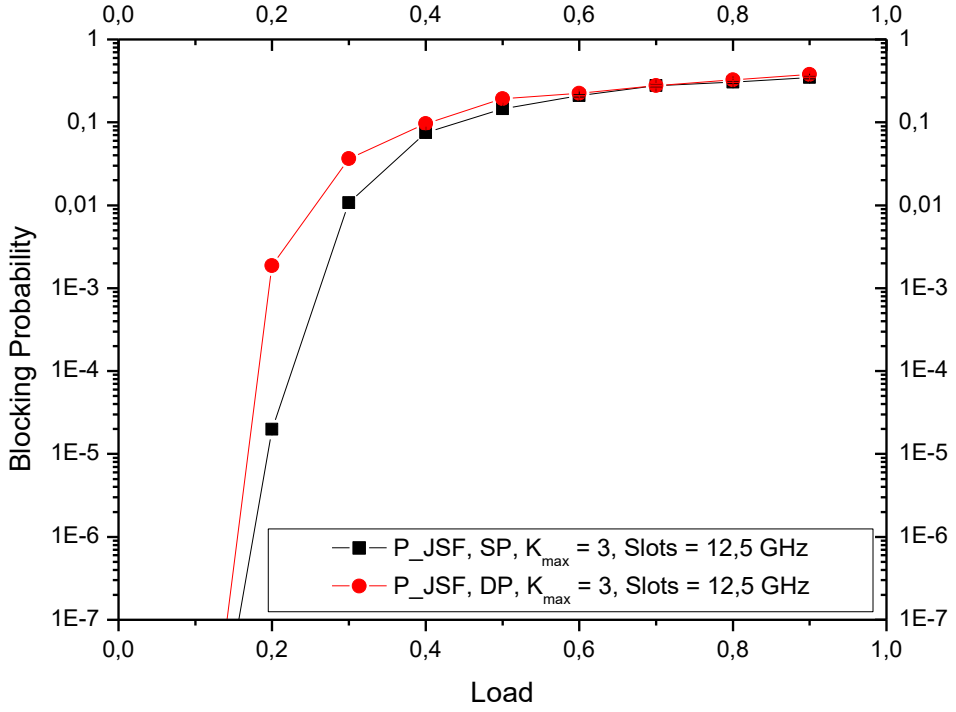


Figura 26. Probabilidad de bloqueo del algoritmo P_JSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 12.5 GHz, K_{max} = 3 y barriendo la carga.

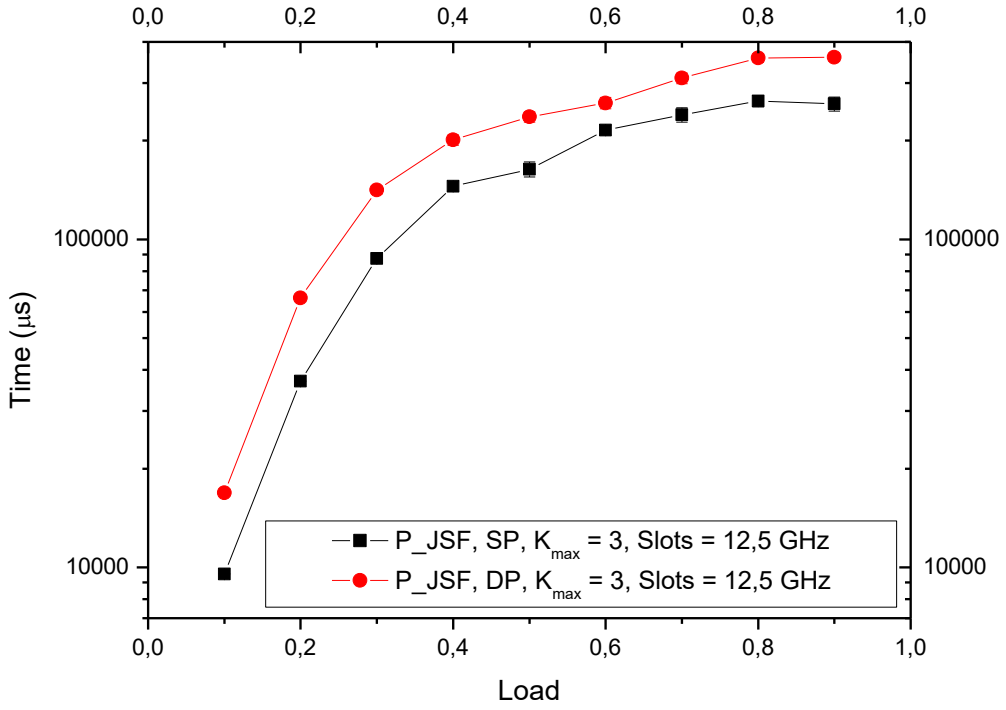


Figura 27. Tiempo de computación del algoritmo P_JSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 12.5 GHz, K_{max} = 3 y barriendo la carga.

Tal y como se observa en la Figura 26 y Figura 27, el algoritmo *shortest-path* (negro) es mucho más eficiente que *least-delayed-path* (rojo), tanto en probabilidad de bloqueo como en tiempo. Estos resultados son también esperados, ya que con *least-delayed-path* estamos dando prioridad al retardo de propagación, parámetro que no tiene ninguna influencia en estas gráficas, con respecto al número de saltos. Por ejemplo, supongamos que entre el nodo A y el nodo B, la ruta con menor número de saltos cuenta con 2, y la ruta que introduce menos retardo tiene 4 saltos. Cuando queramos establecer un *lightpath* entre ellos, en el primer caso se asignará a dos enlaces un ancho de banda determinado, que no podrá ser utilizado por otros *lightpaths* hasta que se libere el *lightpath* en cuestión. En el segundo caso, este ancho de banda es asignado a cuatro enlaces, dos más que en el caso anterior. Este hecho, al contrario que el retardo, sí que influye y de gran manera en la probabilidad de bloqueo resultante, observando la Figura 26. Por tanto, podemos concluir que el algoritmo *shortest-path* será más eficiente que *least-delayed-path* en cualquiera de los algoritmos de asignación que usemos.

El parámetro que nos queda por variar es la protección. En la Figura 28 y Figura 29 mostramos la probabilidad de bloqueo y el tiempo de computación, respectivamente, con la protección desactivada, usando el algoritmo *shortest-path* y barriendo la carga.

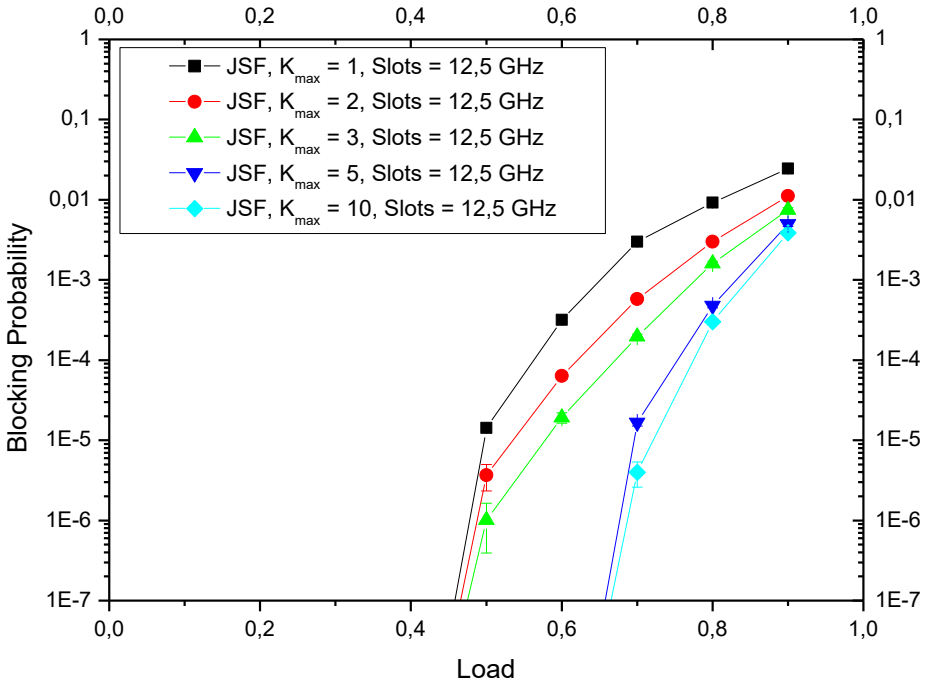


Figura 28. Probabilidad de bloqueo del algoritmo JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path y barriendo la carga.

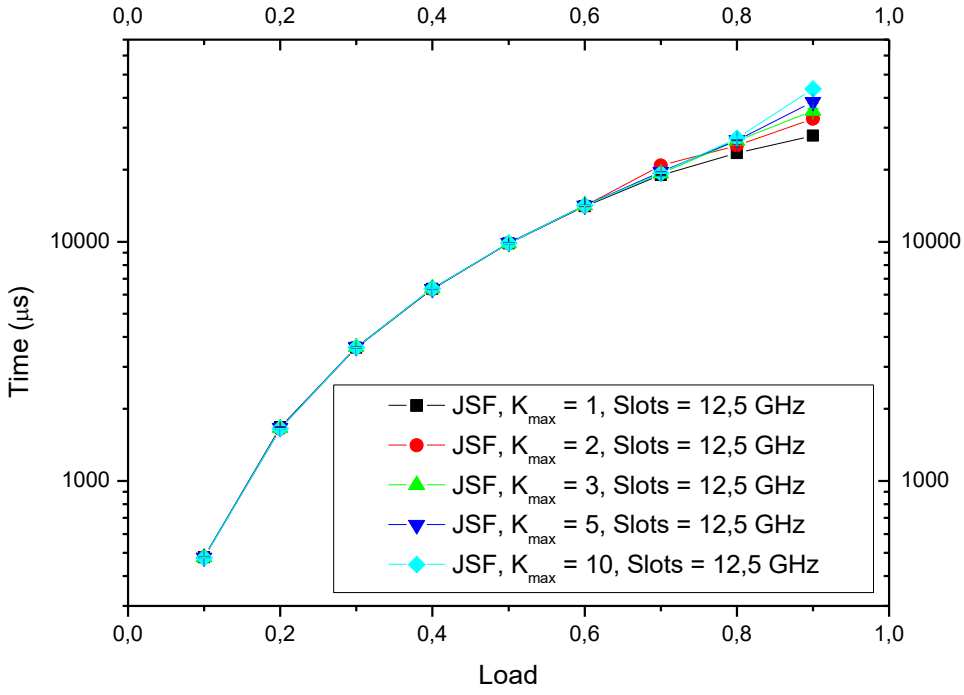


Figura 29. Tiempo de computación del algoritmo JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path y barriendo la carga.

Con estas gráficas podemos obtener varias conclusiones. En primer lugar, la probabilidad de bloqueo es mucho más baja con la protección desactivada, lo cual

es evidente porque en el otro caso reservamos mucho más ancho de banda. Sin embargo, en este caso la robustez ante fallos es mucho menor, y habría que usar técnicas de restauración para solventar posibles problemas en la red. Por otra parte, podemos observar en la Figura 29 que el tiempo de computación es prácticamente idéntico para todos los caminos. Finalmente, se puede ver en la Figura 28 que existe una gran diferencia entre usar como máximo 3 y 5 caminos en cuanto a probabilidad de bloqueo se refiere. Esto podría ser debido a que hubiera enlaces poco utilizados, de modo que en las tres primeras rutas entre cada par de nodos apenas se usen, y que es a partir de 5 caminos cuando se saca provecho de ellos. Aun así, podríamos decir que, en este caso, lo más eficiente sería usar 10 caminos como máximo, ya que el tiempo no varía significativamente, pero la probabilidad de bloqueo sí.

A continuación, mostramos la Figura 30 y Figura 31, que son similares a la Figura 28 y Figura 29, respectivamente, con la diferencia de que, en este caso, usamos el algoritmo *least-delayed-path* en lugar de *shortest-path*. Tal y como se observa, las conclusiones que podemos sacar son equivalentes.

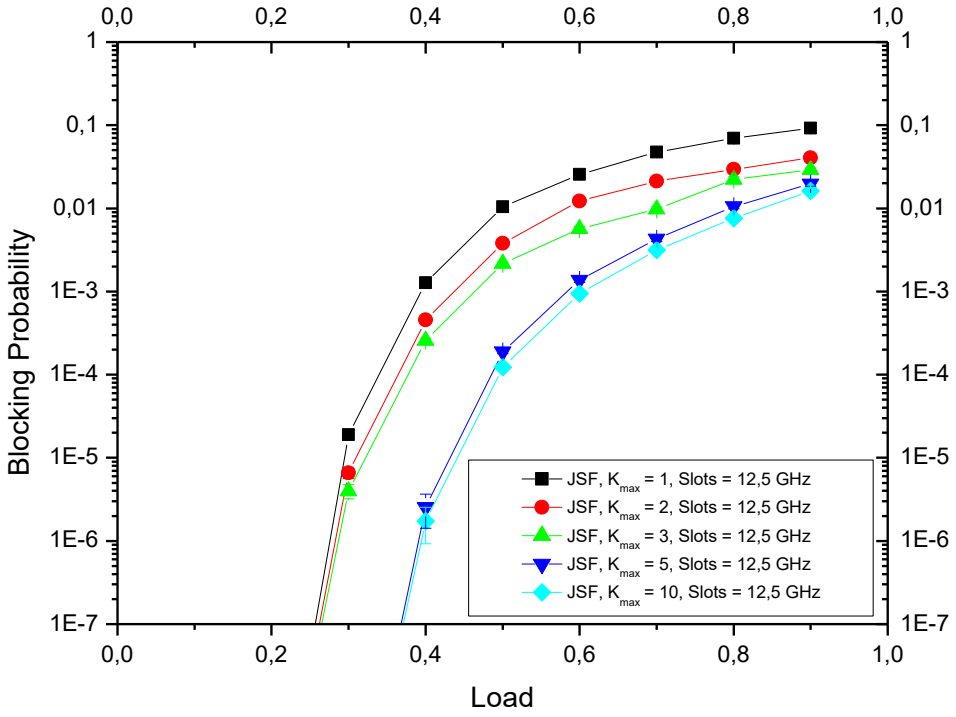


Figura 30. Probabilidad de bloqueo del algoritmo JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según least-delayed-path y barriendo la carga.

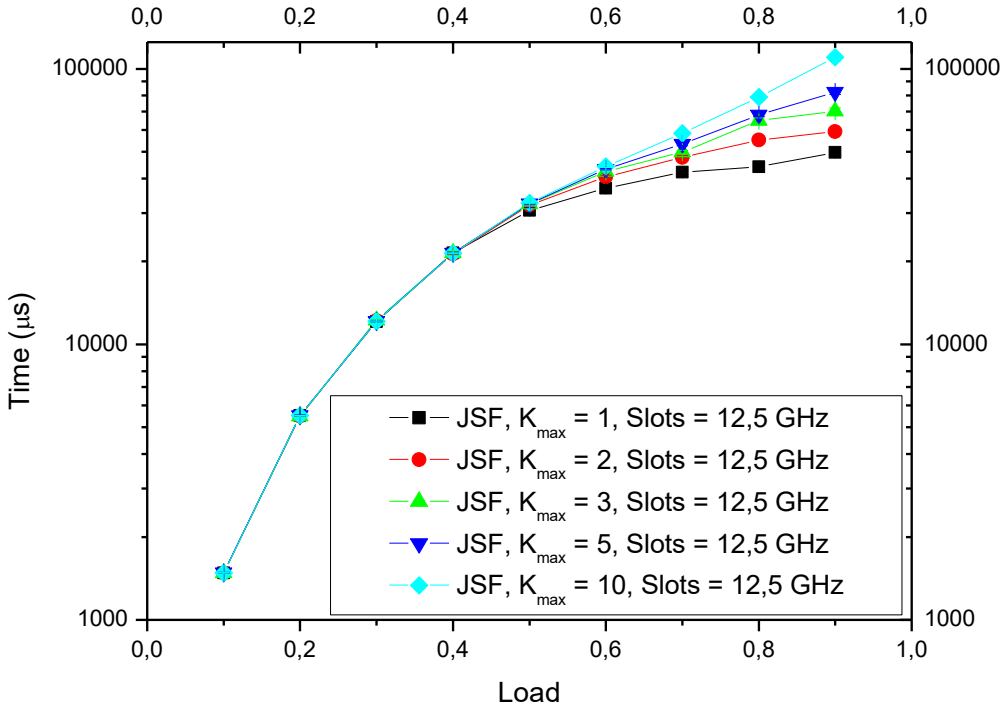


Figura 31. Tiempo de computación del algoritmo JSF-FF para varios K_{max}, con tamaño de slot de 12.5 GHz, ordenación de rutas según least-delayed-path y barriendo la carga.

Por último, vamos a realizar una comparación entre protección activada y desactivada, y entre el uso de *shortest-path* y *least-delayed-path*, todo ello en la misma gráfica, de manera que podamos observar de una manera más idónea todos los resultados. En la Figura 32 y Figura 33 podemos ver la probabilidad de bloqueo y el tiempo, respectivamente, todo ello con 5 caminos, que hemos visto que es más eficiente que 3 en este contexto, y tamaño de *slot* de 12.5 GHz.

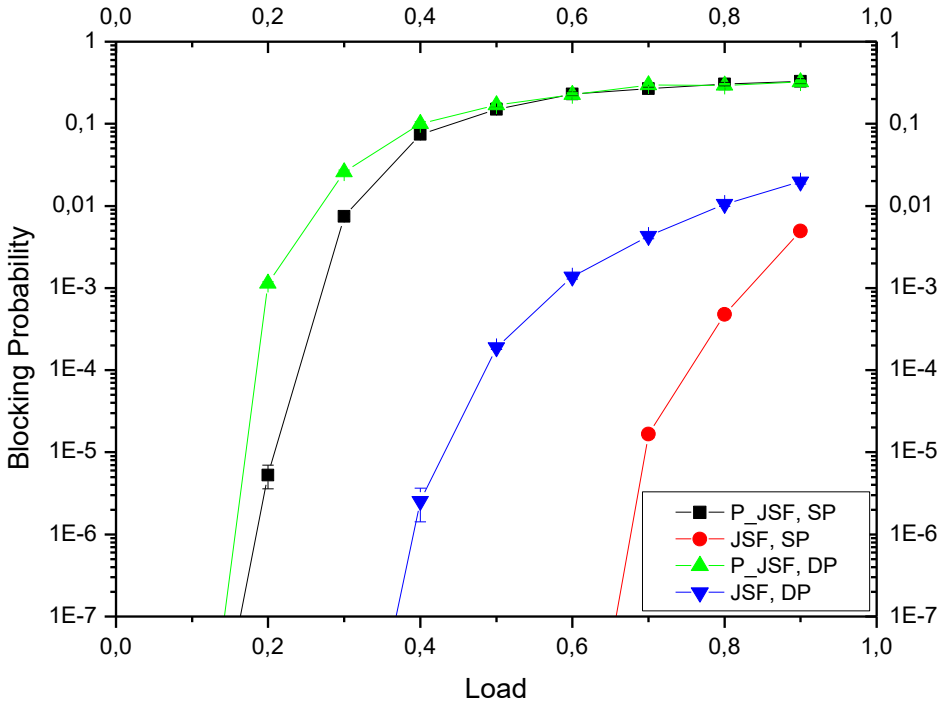


Figura 32. Probabilidad de bloqueo de los algoritmos P_JSF-FF y JSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 12.5 GHz, K_{max} = 5 y barriendo la carga.

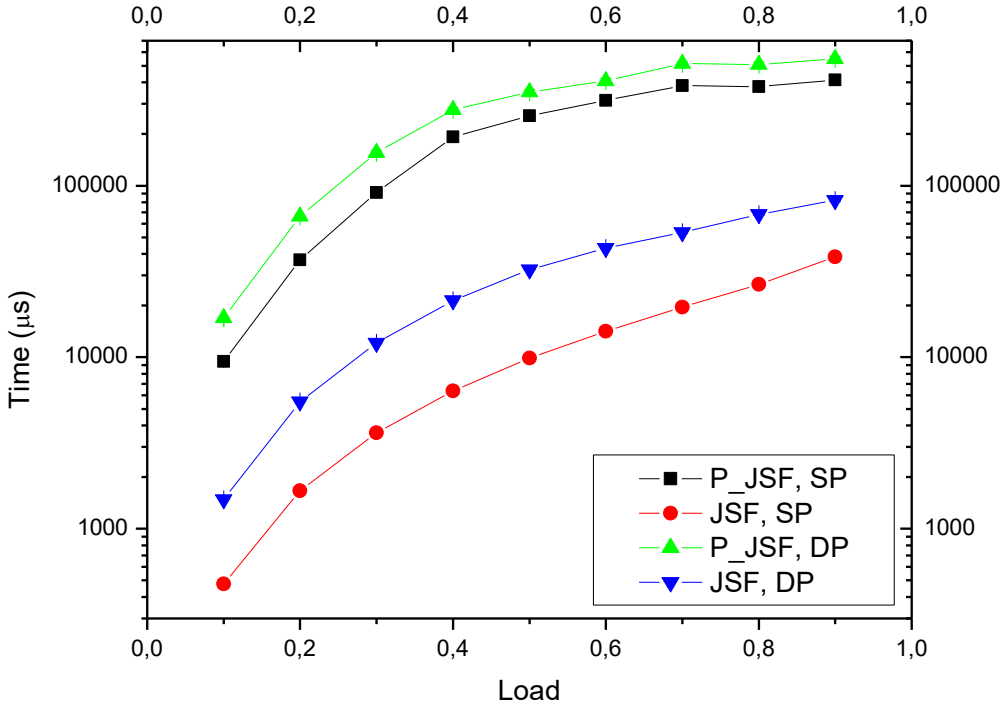


Figura 33. Tiempo de computación de los algoritmos P_JSF-FF y JSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 12.5 GHz, K_{max} = 5 y barriendo la carga.

A la vista de la Figura 32 y Figura 33, llegamos a la conclusión de que *shortest-path* es mucho más eficiente que *least-delayed-path*, independientemente de si usamos rutas de protección o no. Aunque ya se apreciaba en la Figura 28 con respecto a la Figura 20, también se puede decir que sin protección el rendimiento es muy superior, tanto en probabilidad de bloqueo como en tiempo de computación (aproximadamente un orden de magnitud inferior).

4.2. Disjoint Spectrum Fixed-Grid (First-Fit)

Este algoritmo también se basa en *Fixed-Grid*, por lo que vamos a comprobar qué tamaño de *slot* es el más eficiente. Para ello, vamos a realizar una simulación con la protección activada, número de caminos máximo igual a 3, rutas ordenadas según *shortest-path* y barriendo la carga, obteniendo la probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 34 y Figura 35, respectivamente.

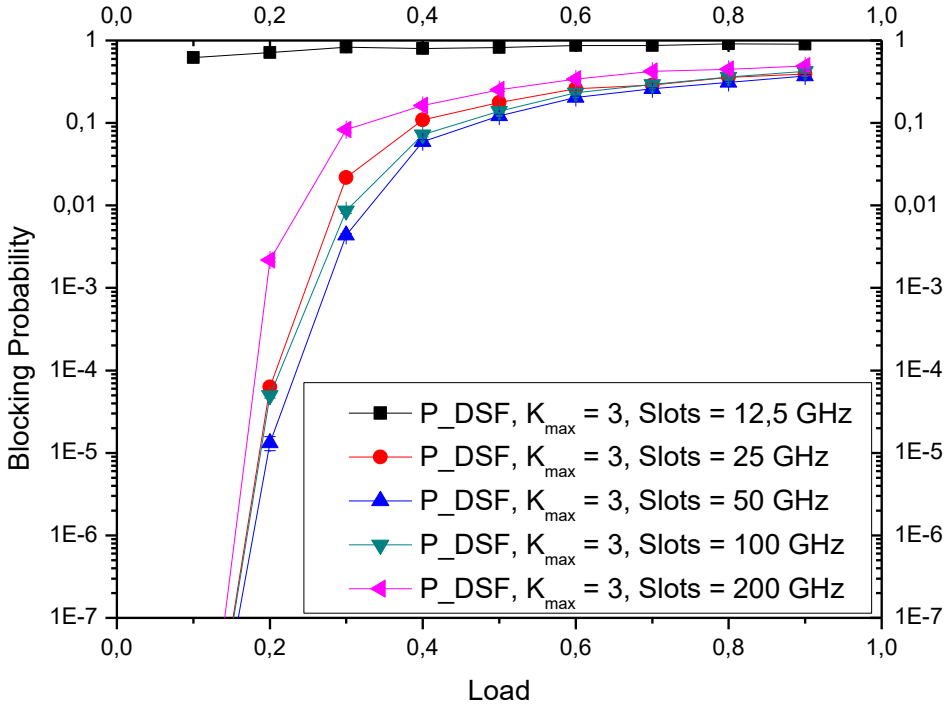


Figura 34. Probabilidad de bloqueo del algoritmo P_DSF-FF para varios tamaños de slot, con K_{max} = 3, ordenación de rutas según shortest-path y barriendo la carga.

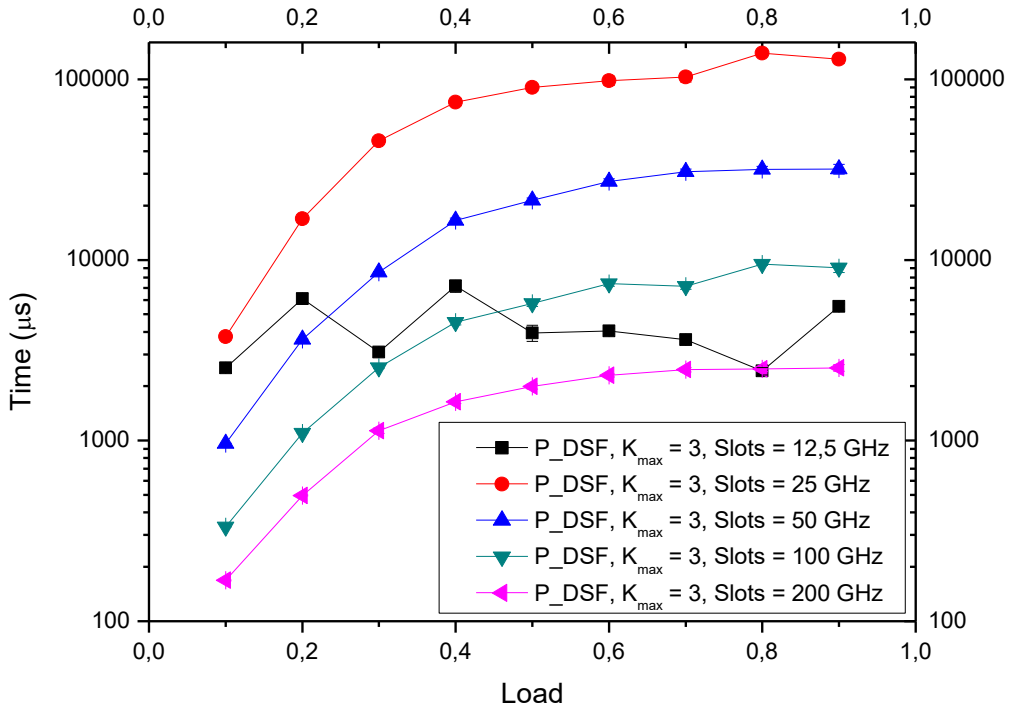


Figura 35. Tiempo de computación del algoritmo P_DSF-FF para varios tamaños de slot, con K_{max} = 3, ordenación de rutas según shortest-path y barriendo la carga.

En la Figura 34 se puede apreciar que se consigue la menor probabilidad de bloqueo con un tamaño de *slot* de 50 GHz, y en la Figura 35 se observa que el menor tiempo de computación se produce con un tamaño de *slot* de 200 GHz, igual que ocurría en el algoritmo P_JSF-FF. Aunque con tamaño de 50 GHz se consigue un tiempo en torno a un orden de magnitud superior, podemos afirmar que el tamaño de *slot* más eficiente será de 50 GHz, ya que buscamos la mínima probabilidad de bloqueo, y será por tanto el que usemos de aquí en adelante.

En este algoritmo no es más eficiente el tamaño de *slot* de 12.5 GHz (como ocurre en P_JSF-FF) porque existe un compromiso entre la precisión y las bandas de guarda, mientras que en P_JSF-FF solo entraba en juego la precisión, ya que siempre se añadía una única banda de guarda. En este caso, en cada grupo de *slots* disjuntos, es decir, en cada grupo de *slots* separados de otro grupo (o simplemente *slots* sueltos), se introducirá una banda de guarda. Quiere decir que, en la situación más extrema, donde todos los *slots* son disjuntos, en todos introduciremos una banda de guarda. El compromiso consiste en qué nos interesa un tamaño lo más grande posible para introducir cuantas menos bandas de guarda, pero a la vez nos interesa un tamaño pequeño para una mejor adaptación a cualquier capacidad sin desperdiciar excesivo ancho de banda. En este compromiso, ya se ha visto en la Figura 34 que el tamaño de *slot* más eficiente es 50 GHz. A través del siguiente ejemplo, se prueba esta conclusión de manera teórica. Suponiendo que queremos asignar una capacidad de 200 GHz, las bandas de guarda son de 10 GHz y todos los *slots* son disjuntos:

- Para un tamaño de *slot* de 12.5 GHz, necesitaremos $ceil\left(\frac{200}{12.5-10}\right) = 80\ slots$, desaprovechando, por tanto, $80 * 12.5 - 200 = 800\ GHz$.
- Para un tamaño de *slot* de 25 GHz, necesitaremos $ceil\left(\frac{200}{25-10}\right) = 14\ slots$, desaprovechando $14 * 25 - 200 = 150\ GHz$.
- Para 50 GHz, necesitaremos 5 $slots$, desaprovechando 50 GHz.
- Para 100 GHz, necesitaremos 3 $slots$, desaprovechando 100 GHz.
- Para 200 GHz, necesitaremos 2 $slots$, desaprovechando 200 GHz.

Pasamos ahora a visualizar la influencia del número máximo de caminos. Para ello, vamos a realizar una simulación con la protección activada, variando el número de caminos, rutas ordenadas según *shortest-path*, con tamaño de *slot* de 50 GHz y barriendo la carga, obteniendo la probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 36 y Figura 37, respectivamente.

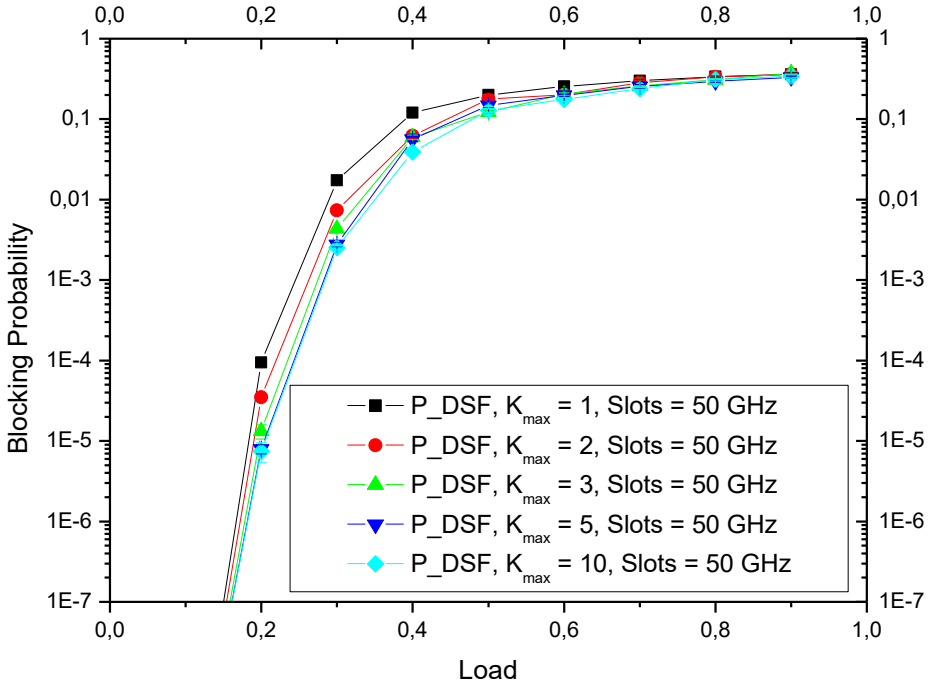


Figura 36. Probabilidad de bloqueo del algoritmo P_DSF-FF para varios K_{max}, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path y barriendo la carga.

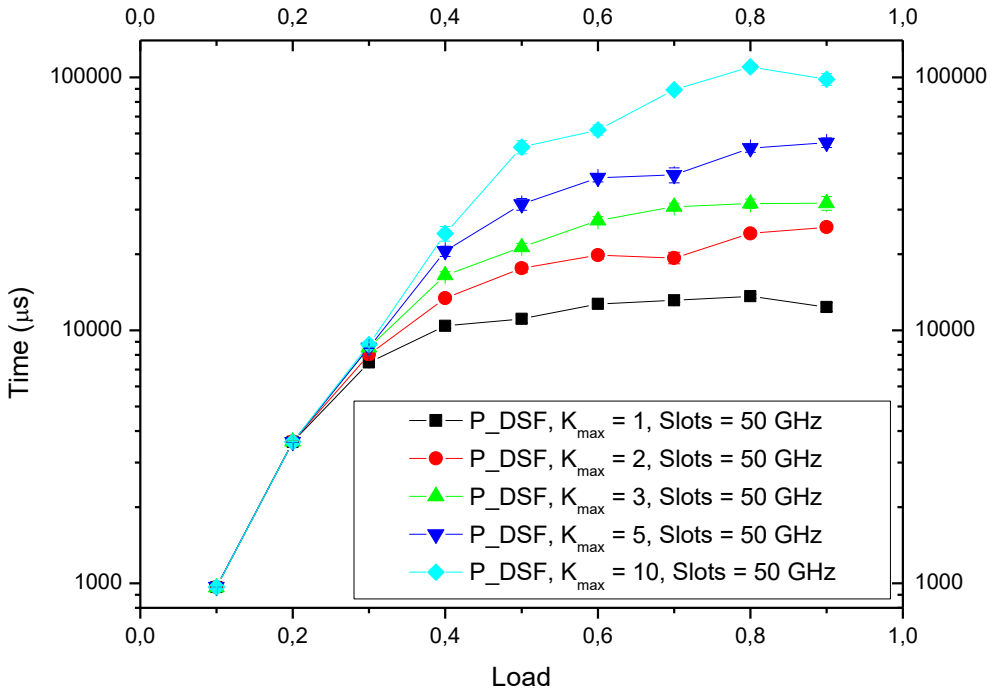


Figura 37. Tiempo de computación del algoritmo P_DSF-FF para varios K_{max}, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path y barriendo la carga.

Al igual que en el algoritmo P_JSF-FF, podemos observar que, en la Figura 36, a partir de 3 caminos máximos, las diferencias son casi inapreciables. En la Figura 37, vemos que según aumenta el número de caminos, aumenta el tiempo medio de computación.

A continuación, en la Figura 38 y Figura 39, mostramos la probabilidad de bloqueo y tiempo, respectivamente, con la protección desactivada, y el resto de parámetros idénticos a los de las gráficas anteriores.

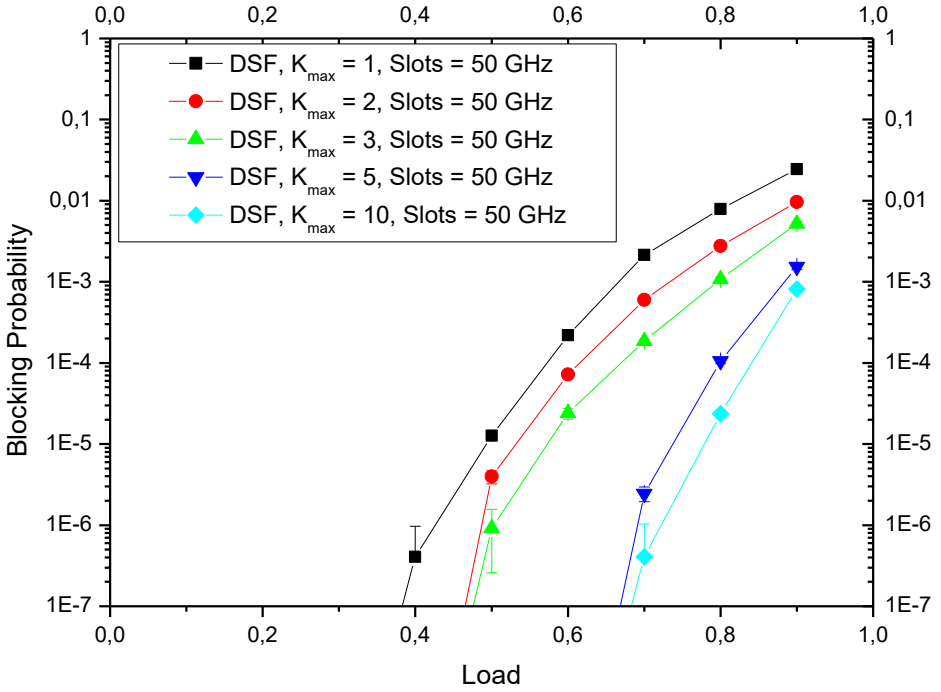


Figura 38. Probabilidad de bloqueo del algoritmo DSF-FF para varios K_{max}, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path y barriendo la carga.

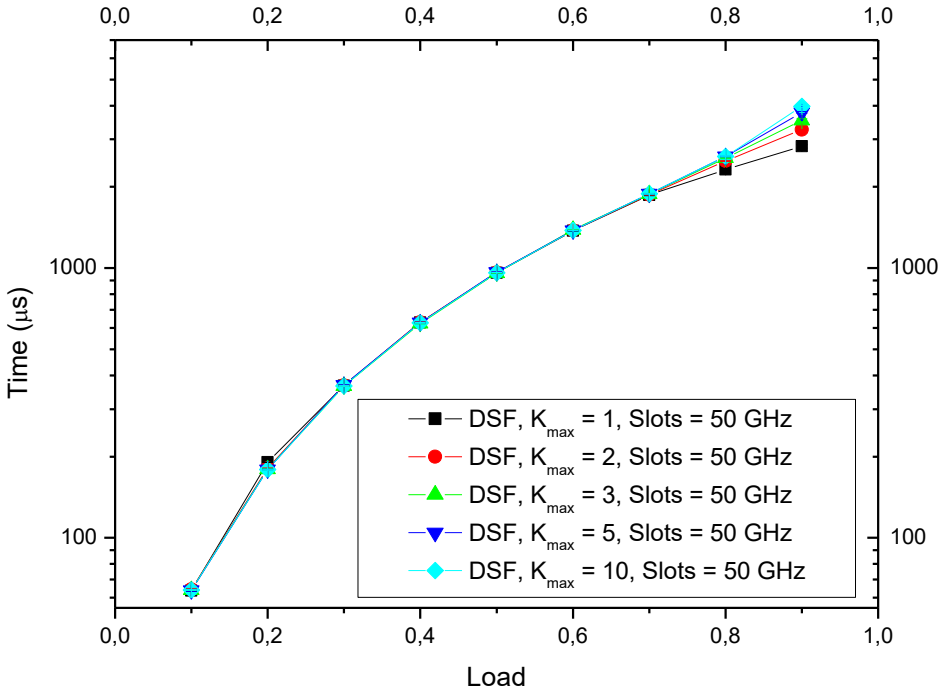


Figura 39. Tiempo de computación del algoritmo DSF-FF para varios K_{max}, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path y barriendo la carga.

Podemos observar que obtenemos unas gráficas muy similares a las del algoritmo JSF-FF.

Para terminar, vamos a realizar una comparación entre protección activada y desactivada, y entre el uso de *shortest-path* y *least-delayed-path*, todo ello en la misma gráfica, de manera que podamos observar de manera más idónea todos los resultados. En la Figura 40 y Figura 41 podemos ver la probabilidad de bloqueo y el tiempo, respectivamente, todo ello con 3 caminos y tamaño de *slot* de 50 GHz.

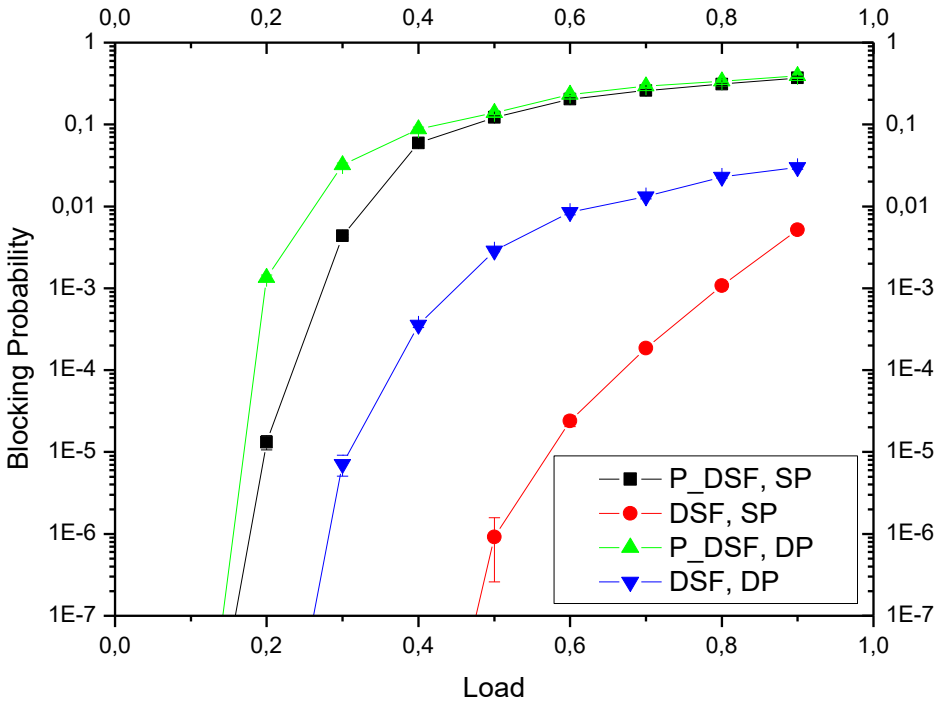


Figura 40. Probabilidad de bloqueo de los algoritmos P_DSF-FF y DSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 50 GHz, K_{max} = 3 y barriendo la carga.

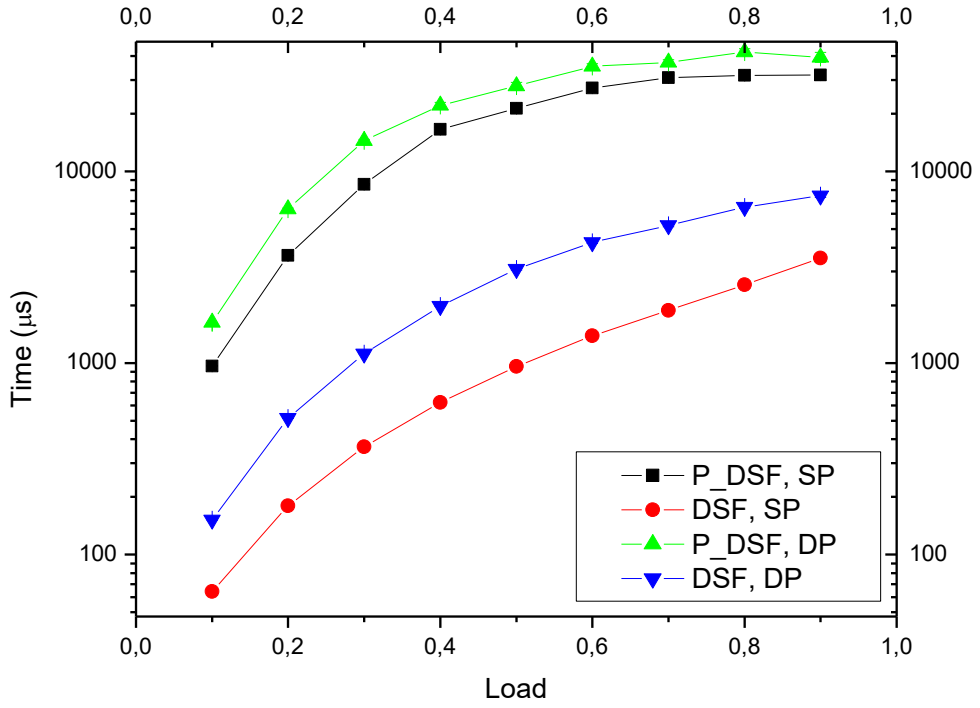


Figura 41. Tiempo de computación de los algoritmos P_DSF-FF y DSF-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con tamaño de slot de 50 GHz, K_{max} = 3 y barriendo la carga.

Tanto en la Figura 40 como en la Figura 41 podemos apreciar la ventaja que supone no reservar rutas de protección (en cuanto a probabilidad de bloqueo y tiempo se refiere), y la diferencia entre *shortest-path* y *least-delayed-path*, proporcionándonos el primero unas prestaciones muy superiores.

4.3. Joint Spectrum Gridless (First-Fit)

En los algoritmos *Gridless* no tenemos *slots*, así que pasamos directamente a comprobar la influencia del número máximo de caminos. Para ello, vamos a realizar una simulación con la protección activada, rutas ordenadas según *shortest-path* y barriendo la carga, obteniendo la probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 42 y Figura 43, respectivamente.

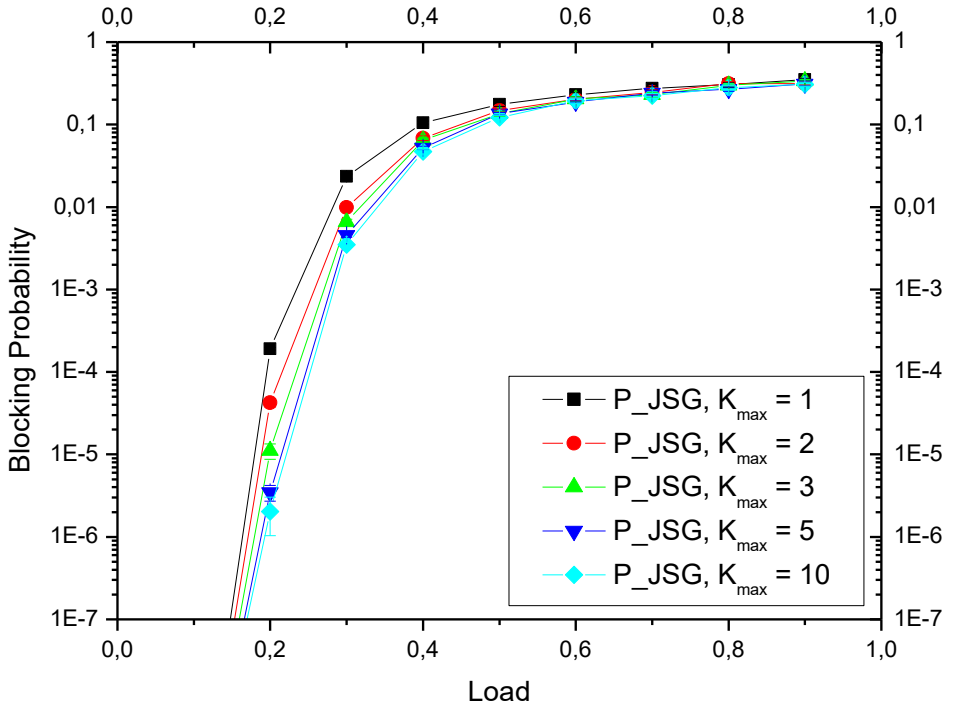


Figura 42. Probabilidad de bloqueo del algoritmo P_JSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

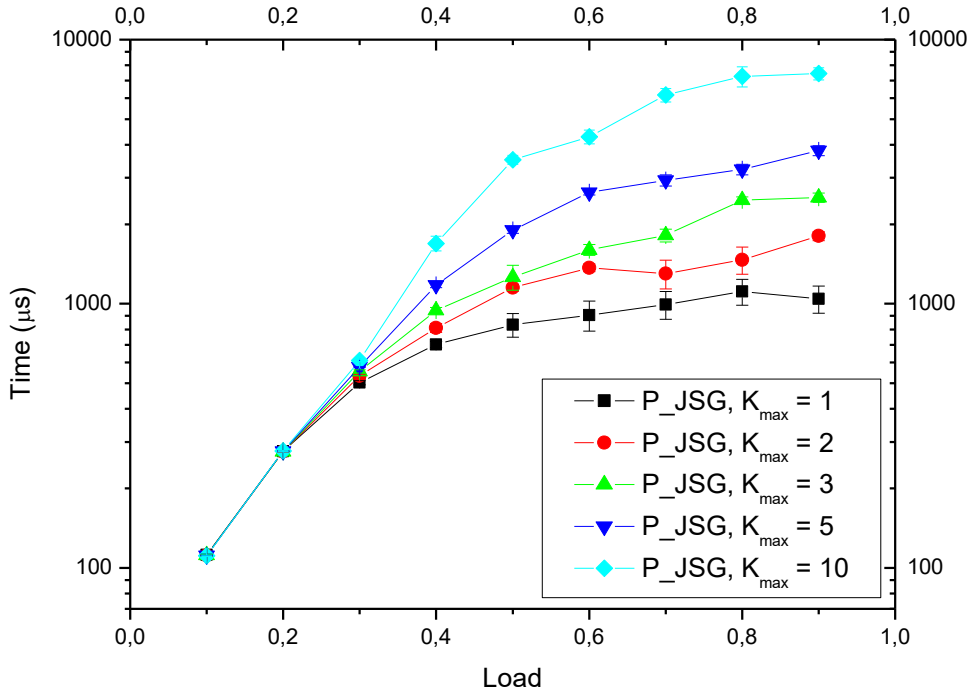


Figura 43. Tiempo de computación del algoritmo P_JSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

Tal y como se aprecia en la Figura 42 y Figura 43, las conclusiones son las mismas que en los algoritmos anteriores. Cuantos más caminos, menor probabilidad de bloqueo, aunque a partir de 5 caminos, ésta apenas mejora. Y cuantos más caminos, mayor tiempo de computación.

La Figura 44 y la Figura 45 son equivalentes a la Figura 42 y Figura 43, respectivamente, con la diferencia de que, en este caso, la protección no está activada. Por tanto, la Figura 44 y Figura 45 se corresponden con la probabilidad de bloqueo y tiempo de computación para *shortest-path*.

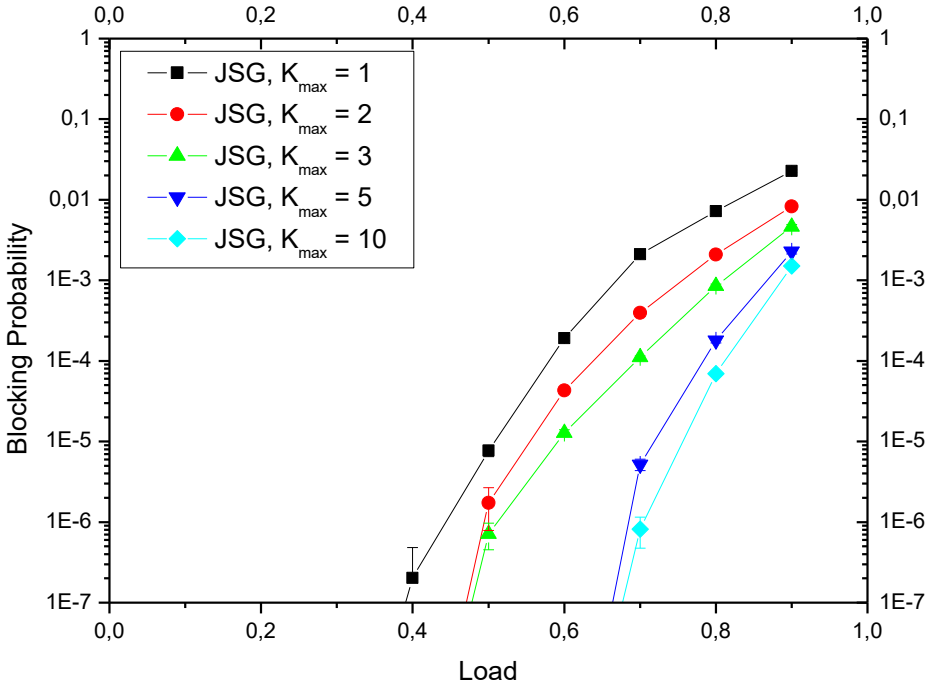


Figura 44. Probabilidad de bloqueo del algoritmo JSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

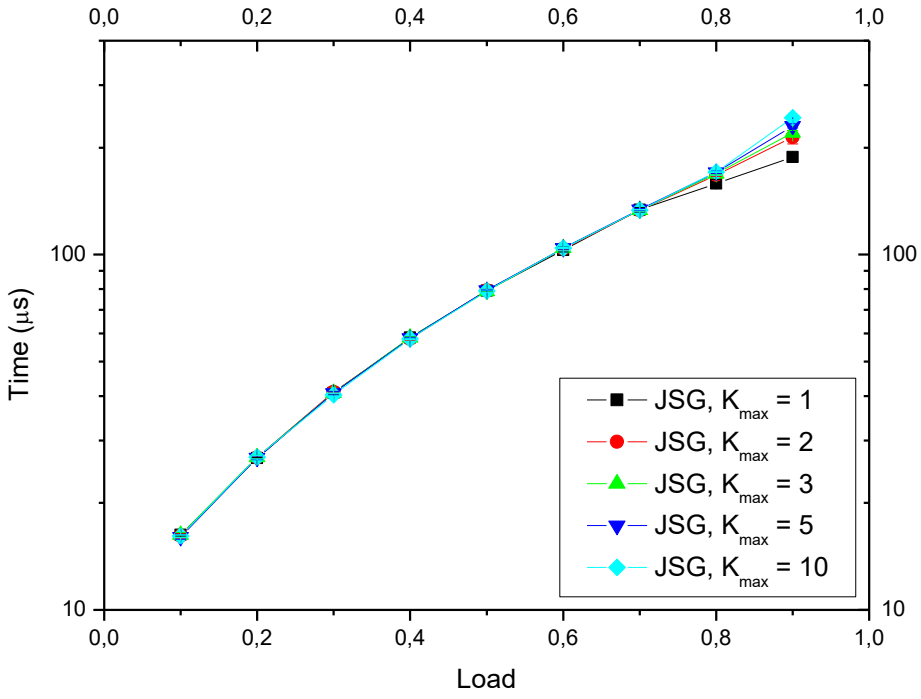


Figura 45. Tiempo de computación del algoritmo JSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

En estas situaciones, podemos observar fenómenos similares a los otros algoritmos ya estudiados, como que el tiempo es prácticamente idéntico para todos los caminos, o la gran diferencia que existe en la probabilidad de bloqueo entre 3 y 5 caminos.

Por último, vamos a realizar una comparación en función del uso de rutas de protección y también con *shortest-path* y *least-delayed-path*, obteniendo probabilidad de bloqueo y tiempo de computación para 3 caminos, mostrados en la Figura 46 y Figura 47, respectivamente.

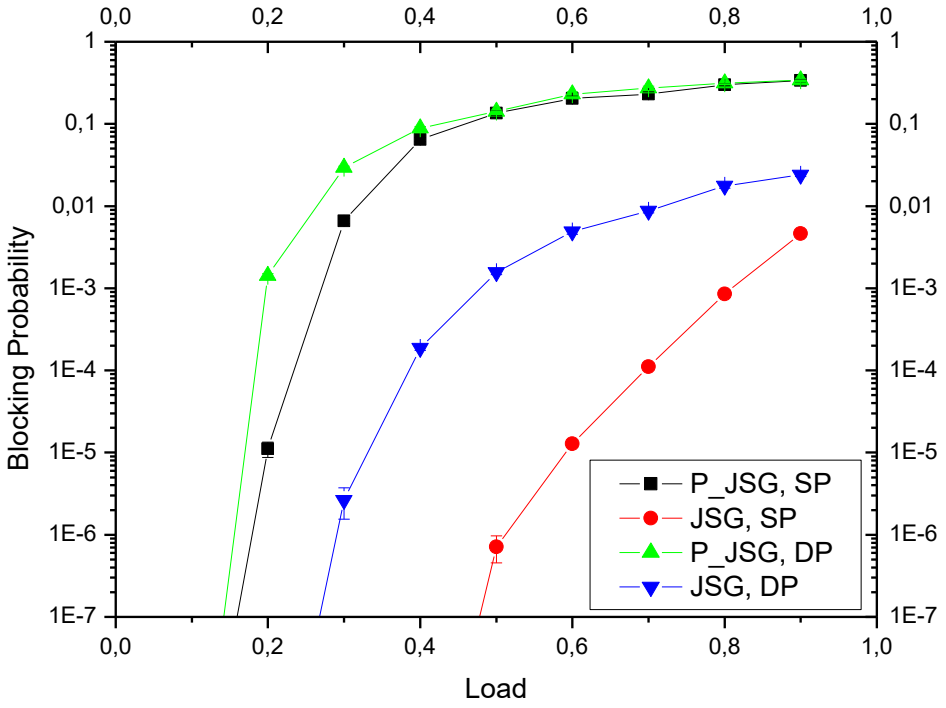


Figura 46. Probabilidad de bloqueo de los algoritmos P_JSG-FF y JSG-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con K_{max} = 3 y barriendo la carga.

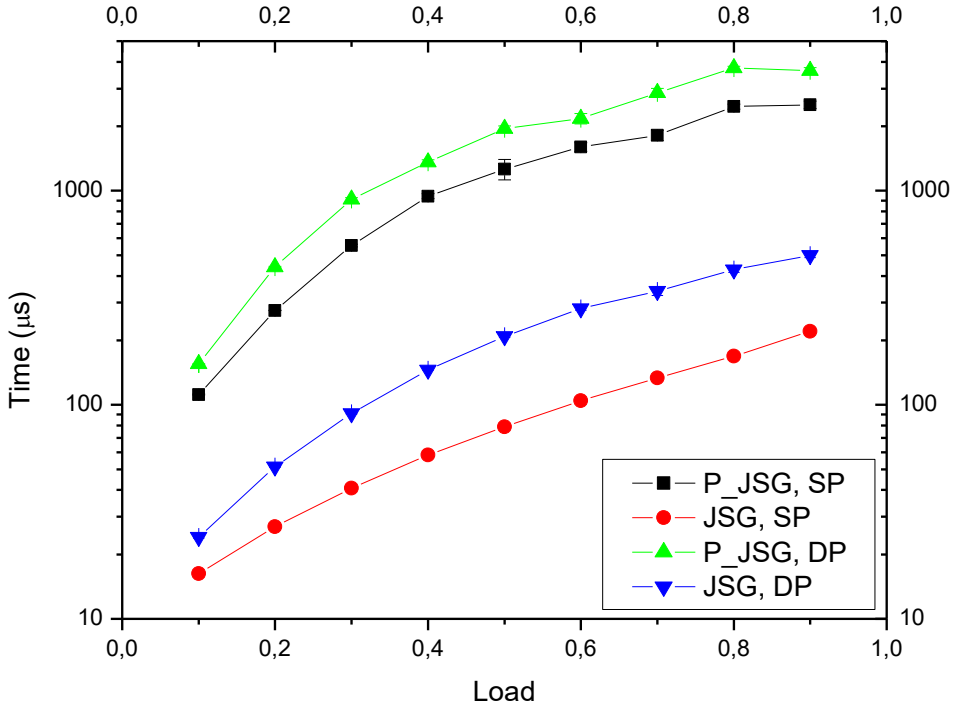


Figura 47. Tiempo de computación de los algoritmos P_JSG-FF y JSG-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con K_{max} = 3 y barriendo la carga.

Tal y como podemos observar en la Figura 46 y Figura 47, los algoritmos con protección desactivada obtienen prestaciones mucho mejores y aquellos que usan *shortest-path* son más eficientes que los que usan *least-delayed-path*, es decir, los comportamientos son parejos a los algoritmos anteriormente estudiados. No obstante, es claro que unos algoritmos ofrecerán características superiores a otros, hecho que es difícil de observar a simple vista. Por ello, al terminar con todos, añadiremos un apartado de comparativa entre algoritmos para ver cuál es más eficiente.

4.4. Disjoint Spectrum Gridless (First-Fit)

Comenzamos comprobando el número de caminos con la protección activada, rutas ordenadas según *shortest-path* y barriendo la carga, obteniendo la probabilidad de bloqueo y el tiempo de computación, tal y como se observa en la Figura 48 y Figura 49, respectivamente.

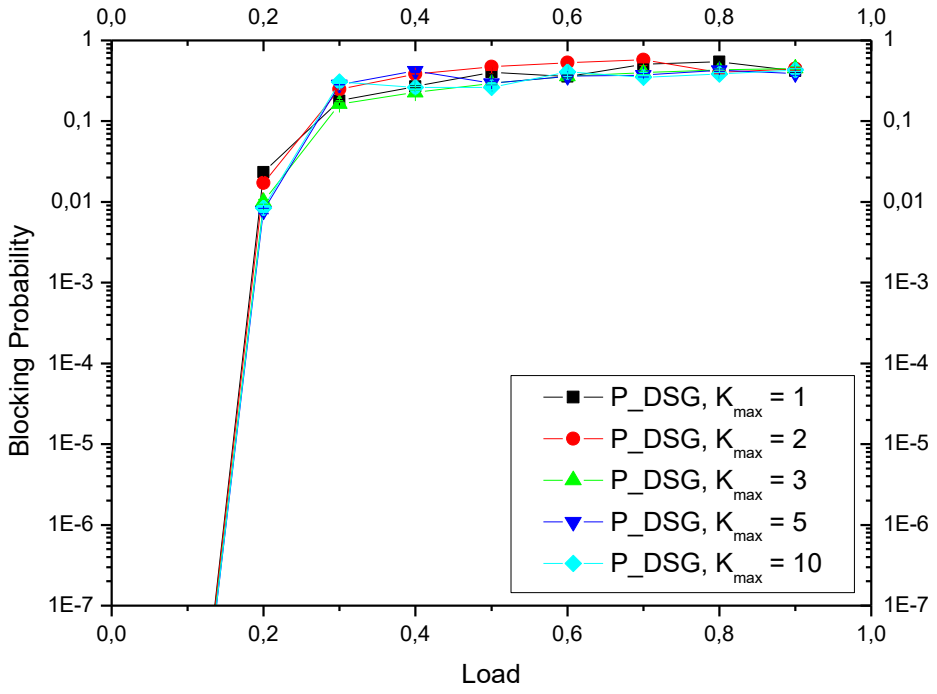


Figura 48. Probabilidad de bloqueo del algoritmo P_DSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

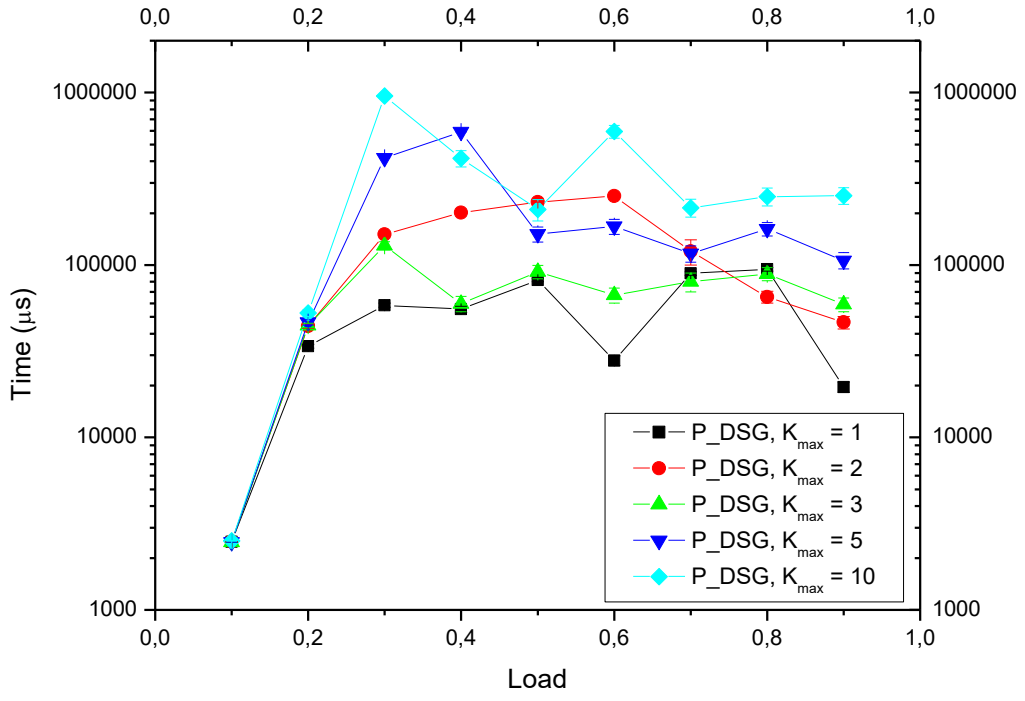


Figura 49. Tiempo de computación del algoritmo P_DSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

Bajo estas condiciones, comenzamos a ver diferencias respecto a los anteriores algoritmos. Podemos observar en la Figura 48 que apenas hay variación entre usar un número máximo de caminos u otro. Por otra parte, en la Figura 49 se aprecia que el tiempo no tiende hacia un valor, como ocurría en casos anteriores, sino que oscila mucho, y sin haber demasiada diferencia entre cada curva. Estos fenómenos se podrían achacar al hecho de que este algoritmo utiliza muchos anchos de banda pequeños, introduciendo muchas bandas de guarda y, por tanto, utilizando un bajo porcentaje de espectro útil. Es por ello que podríamos decir que es un caso similar al de P_DSF-FF de tamaño de *slot* de 12.5 GHz, aunque sin ser tan exagerado. Respecto al número de caminos, probablemente no influyen en demasía porque el bajo uso de espectro.

En la Figura 50 y Figura 51 observamos la probabilidad de bloqueo y tiempo de computación sin el uso de rutas de protección para *shortest-path*.

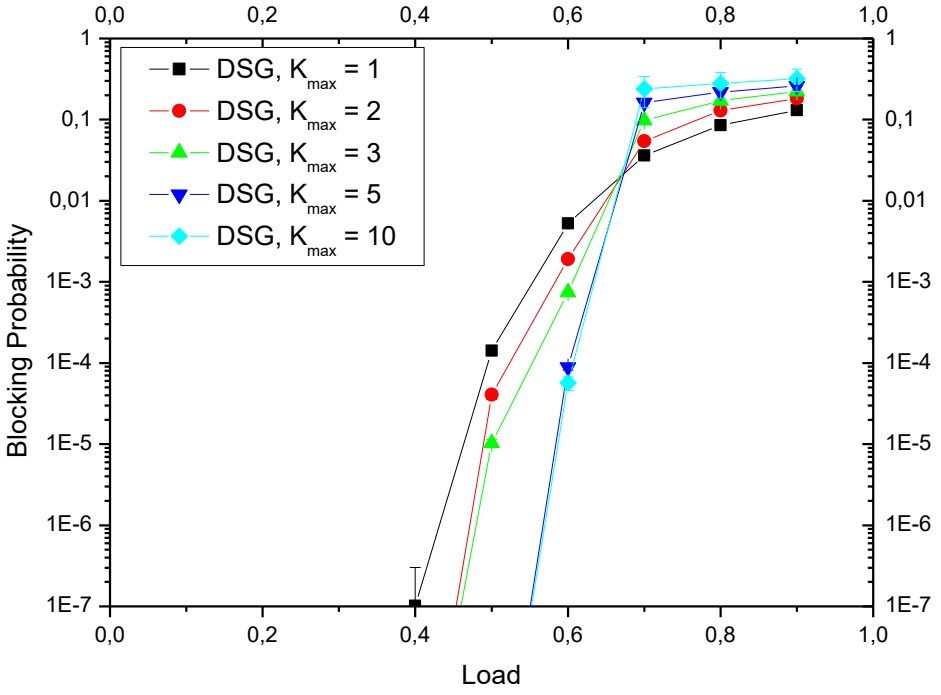


Figura 50. Probabilidad de bloqueo del algoritmo DSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

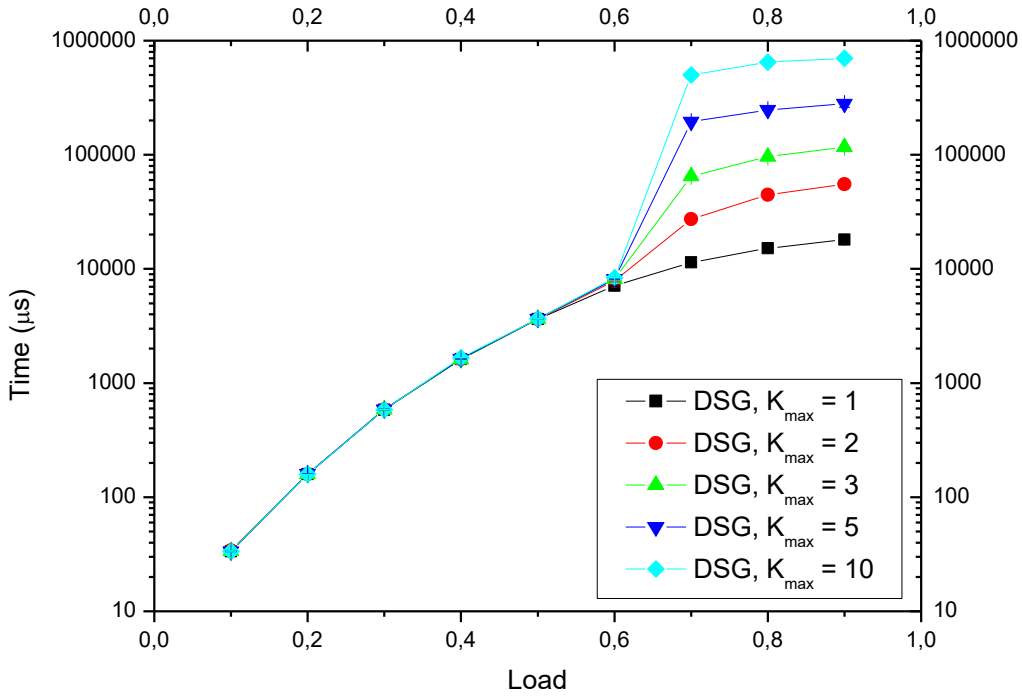


Figura 51. Tiempo de computación del algoritmo DSG-FF para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

En este caso, observamos un comportamiento más normal con respecto a los otros algoritmos. Sí que se aprecia diferencia entre 3 y 5 caminos máximos, aunque no tan pronunciado como en los otros algoritmos, y el tiempo no es tan oscilante.

Por último, vamos a realizar una comparación en función del uso de rutas de protección y también con *shortest-path* y *least-delayed-path*, obteniendo probabilidad de bloqueo y tiempo de computación para 3 caminos, mostrados en la Figura 52 y Figura 53, respectivamente.

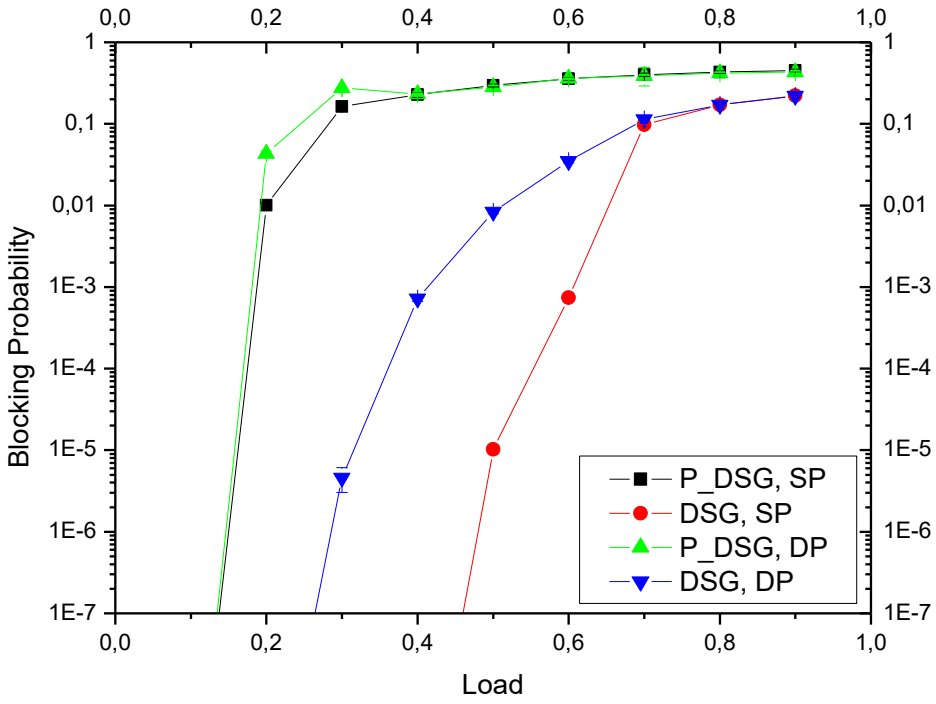


Figura 52. Probabilidad de bloqueo de los algoritmos P_DSG-FF y DSG-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con K_{max} = 3 y barriendo la carga.

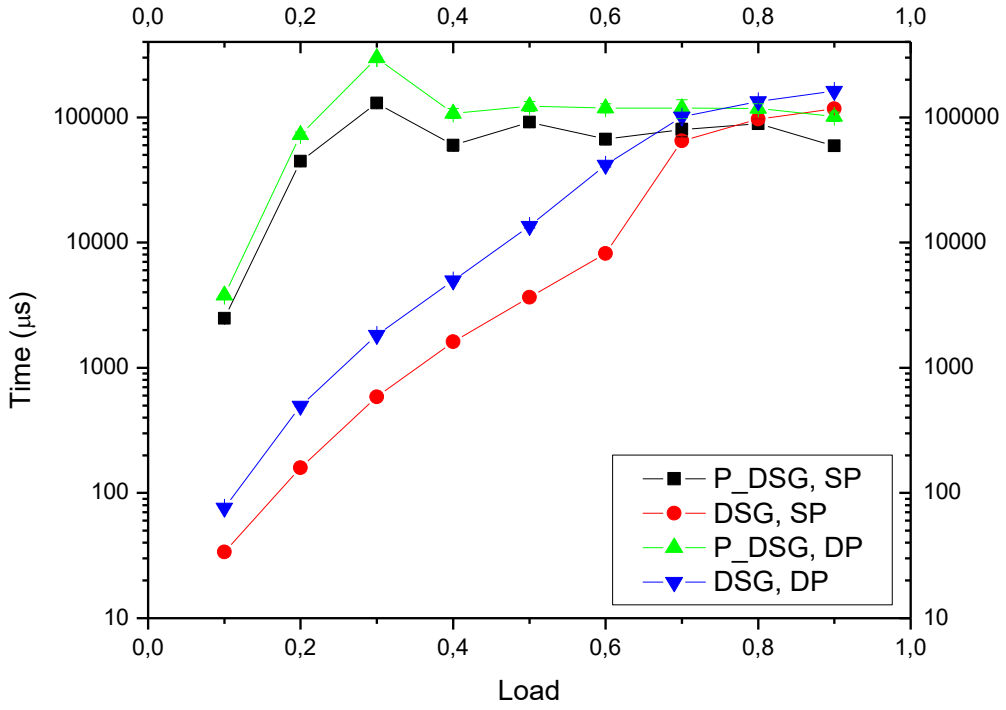


Figura 53. Tiempo de computación de los algoritmos P_DSG-FF y DSG-FF con ordenación de rutas según shortest-path (SP) y least-delayed-path (DP), con K_{max} = 3 y barriendo la carga.

Tal y como podemos observar en la Figura 52 y Figura 53, el comportamiento de este algoritmo es inferior en cuanto a prestaciones con respecto a los algoritmos anteriormente estudiados. No obstante, a continuación mostramos un apartado de comparativa entre algoritmos para ver las diferencias entre ellos.

4.5. Comparativa Algoritmos First-Fit

Pasamos a realizar una comparativa entre los algoritmos *first-fit* para determinar cuál es más eficiente. Haremos por separado los ordenados según *shortest-path* y *least-delayed-path*, puesto que ya sabemos que aquellos con *shortest-path* serán más eficientes. No obstante, conocer qué algoritmo es más eficiente para rutas ordenadas según *least-delayed-path* puede ser importante si queremos utilizarlo para alguna aplicación en la que prima un retardo mínimo.

Comenzamos con los algoritmos que usan *shortest-path*. La Figura 54, Figura 55 y Figura 56 son la comparativa entre los cuatro algoritmos con la protección activada y centrándonos en la probabilidad de bloqueo cuando variamos la carga y el número de caminos máximo, K_{max}.

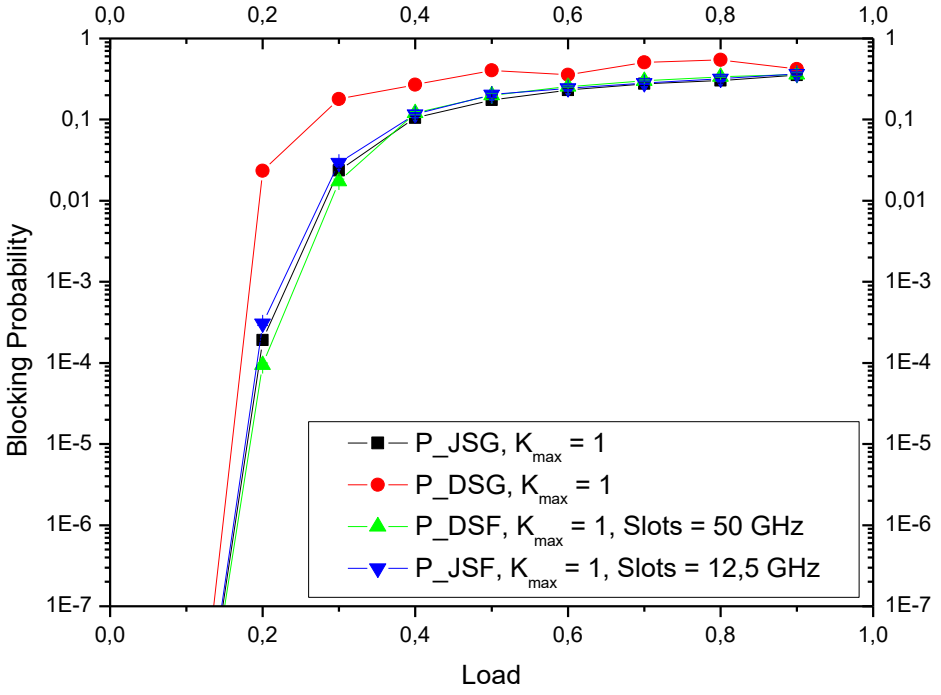


Figura 54. Probabilidad de bloqueo de los algoritmos P_JSG-FF, P_DSG-FF, P_DSF-FF (tamaño de slot de 50 GHz) y P_JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 1 y barriendo la carga.

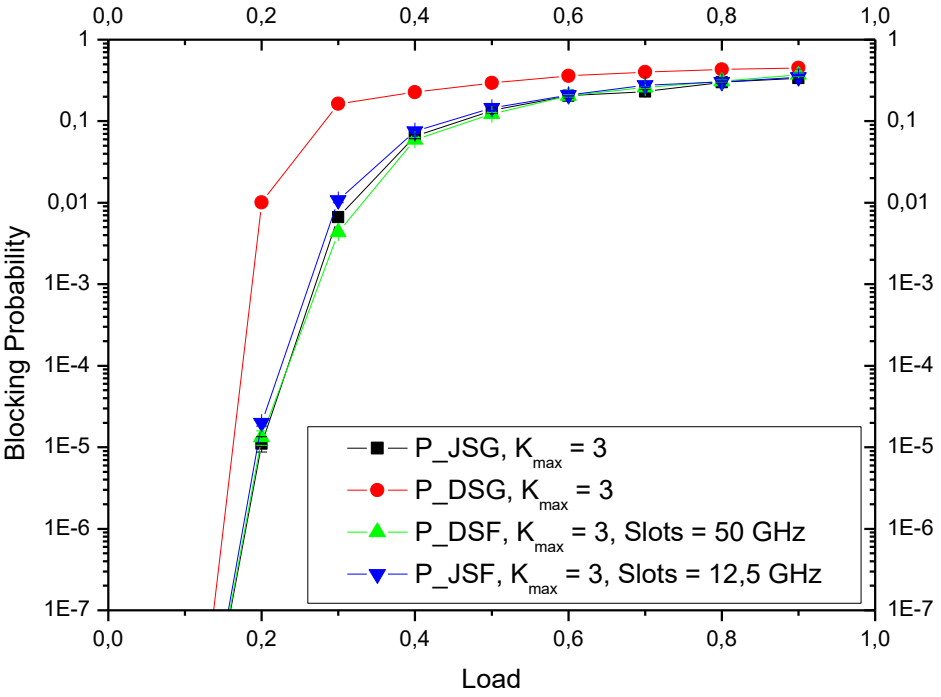


Figura 55. Probabilidad de bloqueo de los algoritmos P_JSG-FF, P_DSG-FF, P_DSF-FF (tamaño de slot de 50 GHz) y P_JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

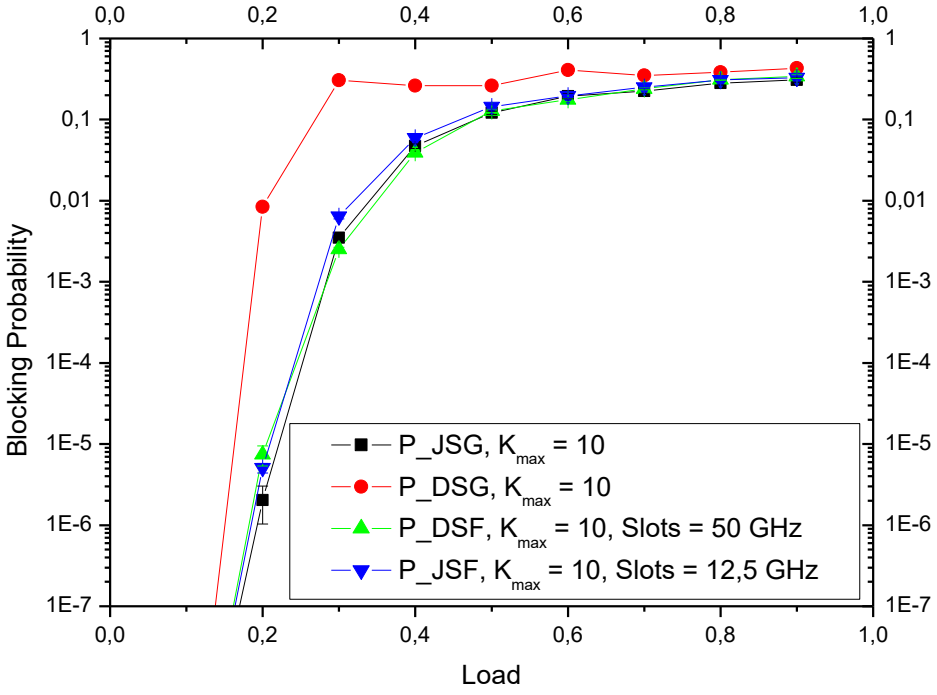


Figura 56. Probabilidad de bloqueo de los algoritmos P_JSG-FF, P_DSG-FF, P_DSF-FF (tamaño de slot de 50 GHz) y P_JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 10 y barriendo la carga.

Las conclusiones son claras. El algoritmo P_DSG-FF está muy por debajo en cuanto a prestaciones respecto al resto de algoritmos. Esto puede ser debido, entre otras cosas, al espectro útil usado por cada algoritmo. En la Figura 57, observamos como el espectro útil de P_DSG-FF, para una carga de 0.2 y barriendo el número de caminos, está muy por debajo del resto. Por otra parte, vemos en la Figura 54 que con un camino el algoritmo más eficiente es P_DSF-FF. Sin embargo, según va aumentando el número de caminos máximo, los algoritmos *joint* pasan a ser más eficientes. Esto se debe a que los algoritmos *disjoint* son más flexibles que los *joint*, a costa de introducir más bandas de guarda. Esta flexibilidad resulta muy beneficiosa con pocos caminos, ya que hay que asignar un ancho de banda en una o dos rutas, pero según aumentan los caminos, resulta contraproducente, pues sobrecarga el espectro, existiendo otras rutas con más frecuencias disponibles. De hecho, en la Figura 56, con un número máximo de caminos igual a 10, vemos que los algoritmos más eficientes se corresponden con los que mejor aprovechan el espectro, tal y como se muestra en la Figura 57.

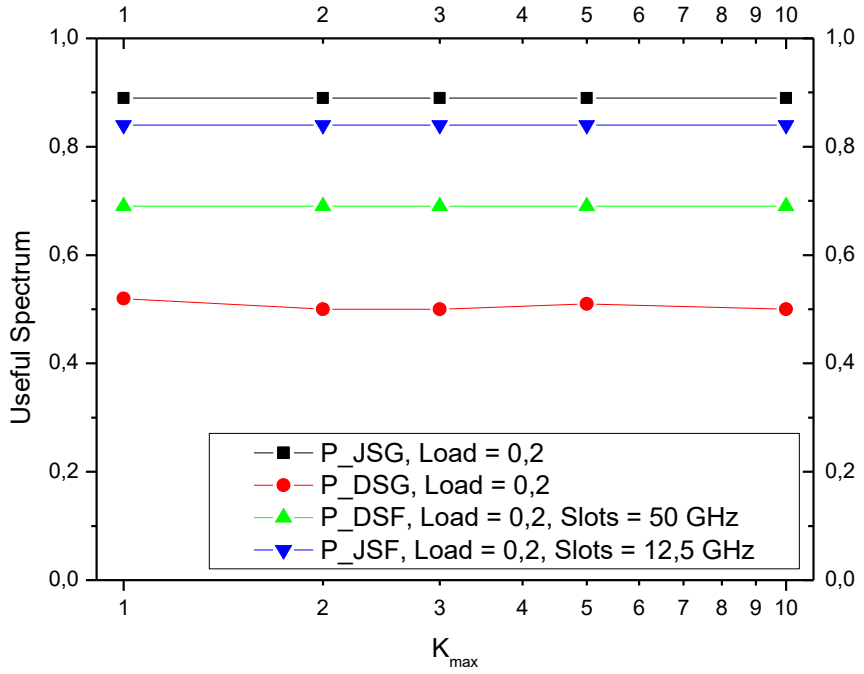


Figura 57. Espectro útil de los algoritmos P_JSG-FF, P_DSG-FF, P_DSF-FF (tamaño de slot de 50 GHz) y P_JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, carga de 0.2 y barriendo K_{max}.

Pasamos ahora a compararlos en función del tiempo medio de computación. Como todas las gráficas son muy parecidas, mostramos solo la referente a 3 caminos (Figura 58), para no sobrecargar el documento con gráficas prácticamente idénticas.

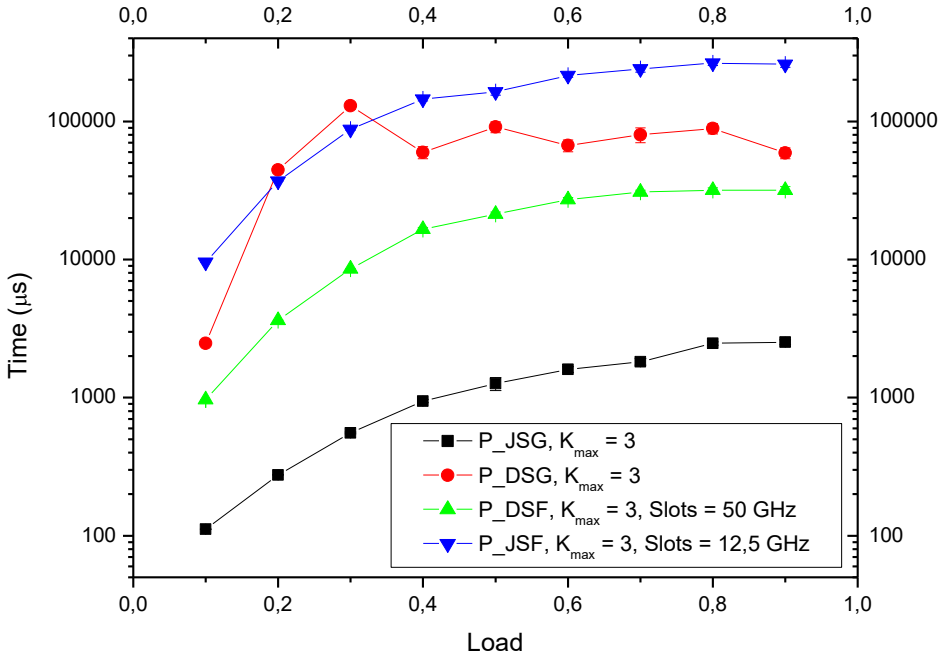


Figura 58. Tiempo de computación de los algoritmos P_JSG-FF, P_DSG-FF, P_DSF-FF (tamaño de slot de 50 GHz) y P_JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

Podemos observar que el algoritmo más eficiente en cuanto a tiempo es P_JSG-FF, seguido a un orden de magnitud por el algoritmo P_DSF-FF. Esto se debe a que P_JSG-FF es el algoritmo más simple y, por tanto, resulta el más rápido. Por otra parte, P_DSG-FF es bastante complejo, lo que redunda en el tiempo de computación. En cuanto a los algoritmos *fixed-grid*, el tiempo no es eficiente porque usan tamaños de *slot* pequeños (12.5 y 50 GHz), y tal y como vimos en sus respectivos apartados, el tamaño de *slot* más eficiente en tiempo era de 200 GHz.

Mostramos a continuación la comparativa entre los algoritmos con la protección desactivada. La Figura 59 y Figura 60 se corresponden con la probabilidad de bloqueo.

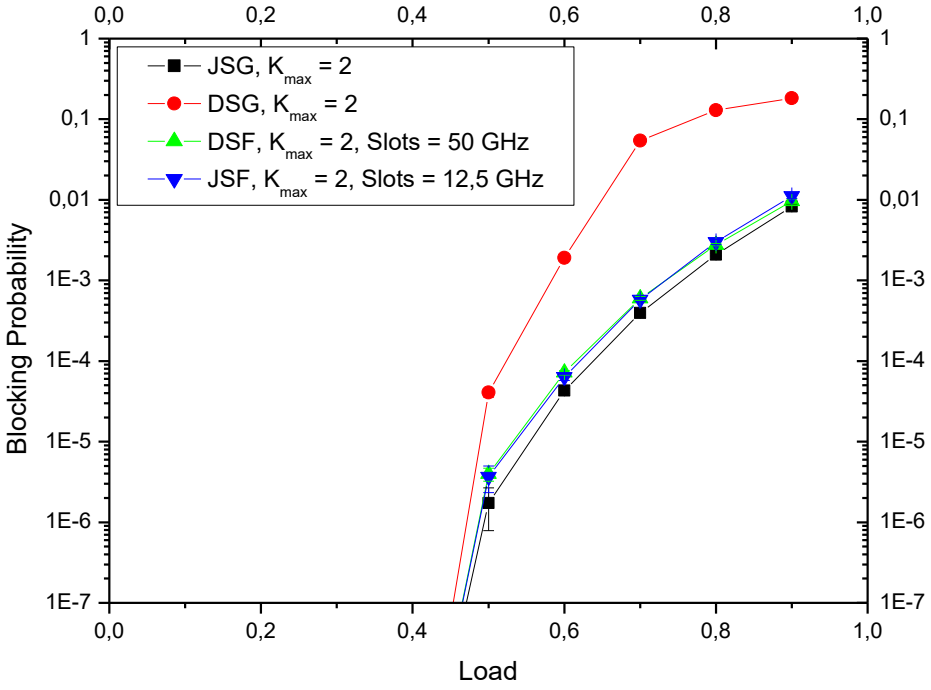


Figura 59. Probabilidad de bloqueo de los algoritmos JSG-FF, DSG-FF, DSF-FF (tamaño de slot de 50 GHz) y JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, $K_{max} = 2$ y barriendo la carga.

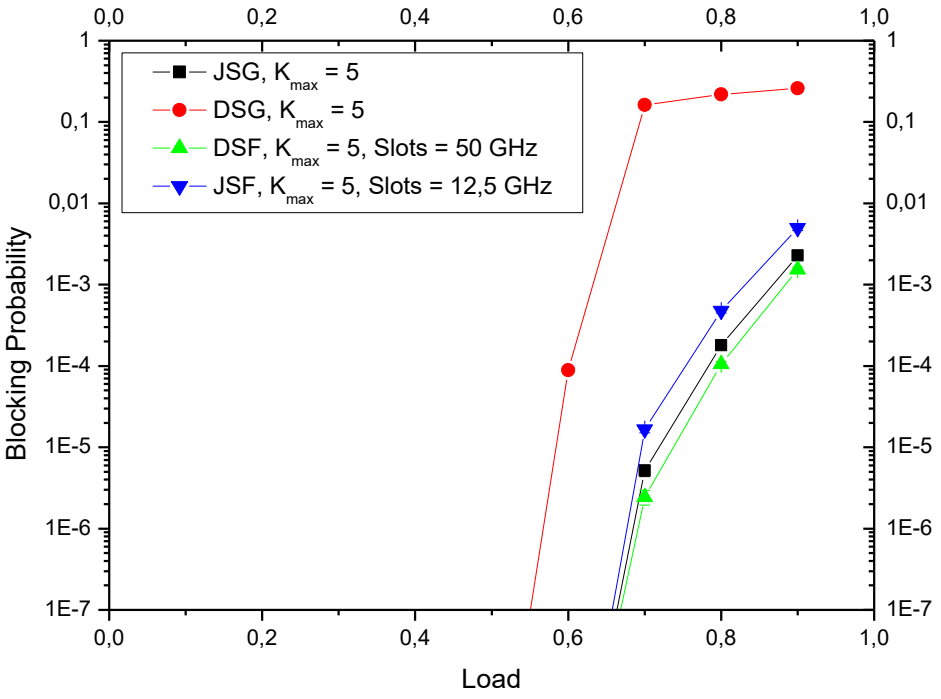


Figura 60. Probabilidad de bloqueo de los algoritmos JSG-FF, DSG-FF, DSF-FF (tamaño de slot de 50 GHz) y JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 5 y barriendo la carga.

Ahora vemos un fenómeno totalmente opuesto al caso anterior. Sin la protección activada, los algoritmos *joint* son más eficientes con un número bajo de caminos. No obstante, cuando estos aumentan, el algoritmo DSF-FF pasa a ser más eficiente. Una posible respuesta a esta coyuntura es que el tráfico total demandado es tan pequeño en comparación con el espectro que la flexibilidad del algoritmo *disjoint* no es relevante. Sin embargo, cuando aumentan los caminos, las nuevas rutas tendrán rutas parcialmente ocupadas, por lo que la flexibilidad pasa a ser importante a la hora de asignar anchos de banda en lugares remotos y evitar el bloqueo. Es decir, en situaciones de mínima y máxima congestión, los algoritmos *joint* son más eficientes, pero cuando la congestión es intermedia, es el DSF-FF el más apropiado.

En cuanto al tiempo, la Figura 61 representa lo que ocurre no solo para 2 caminos, sino que también para el resto, puesto que para todos los caminos las gráficas son semejantes.

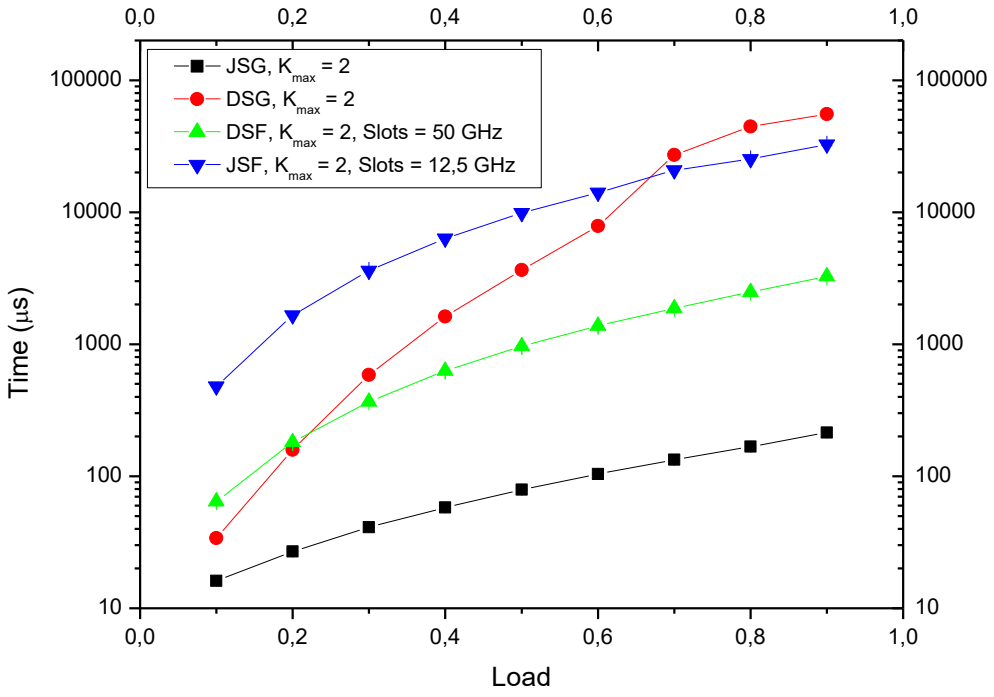


Figura 61. Tiempo de computación de los algoritmos JSG-FF, DSG-FF, DSF-FF (tamaño de slot de 50 GHz) y JSF-FF (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 2 y barriendo la carga.

Al igual que en la situación con la protección activada, el algoritmo JSG-FF sigue siendo el más eficiente.

Ya que no se aprecian grandes diferencias entre las probabilidades de bloqueo de JSG-FF y DSF-FF, pero el algoritmo JSG-FF es mucho más rápido (ídem con la protección activada), podemos concluir que los mejores algoritmos con la filosofía *first-fit* serán P_JSG-FF y JSG-FF.

Respecto a *least-delayed-path*, obtenemos unas gráficas muy similares, por lo que no las mostramos en el documento. No obstante, las conclusiones que se pueden sacar son las mismas, es decir, en tiempo el mejor algoritmo es JSG-FF, y en probabilidad de bloqueo, dependiendo del estado del espectro, en unos casos es mejor DSF-FF y en otros JSG-FF.

4.6. Disjoint Spectrum Gridless (Best-Gap)

Debido a que las gráficas y conclusiones a sacar del resto de algoritmos *best-gap* son prácticamente iguales a los ya descritos con *first-fit*, únicamente vamos a destacar el que cambia significativamente, DSG-BG. No obstante, tras este apartado, haremos una comparación entre todos los algoritmos *best-gap* y así ver cuál es más eficiente. Después se hará una comparación entre *best-gap* y *first-fit*.

Respecto a P_DSG-BG, en la Figura 62 y Figura 63 mostramos la probabilidad de bloqueo y tiempo de computación barriendo el número de caminos. Además, las rutas están ordenadas según el algoritmo *shortest-path*.

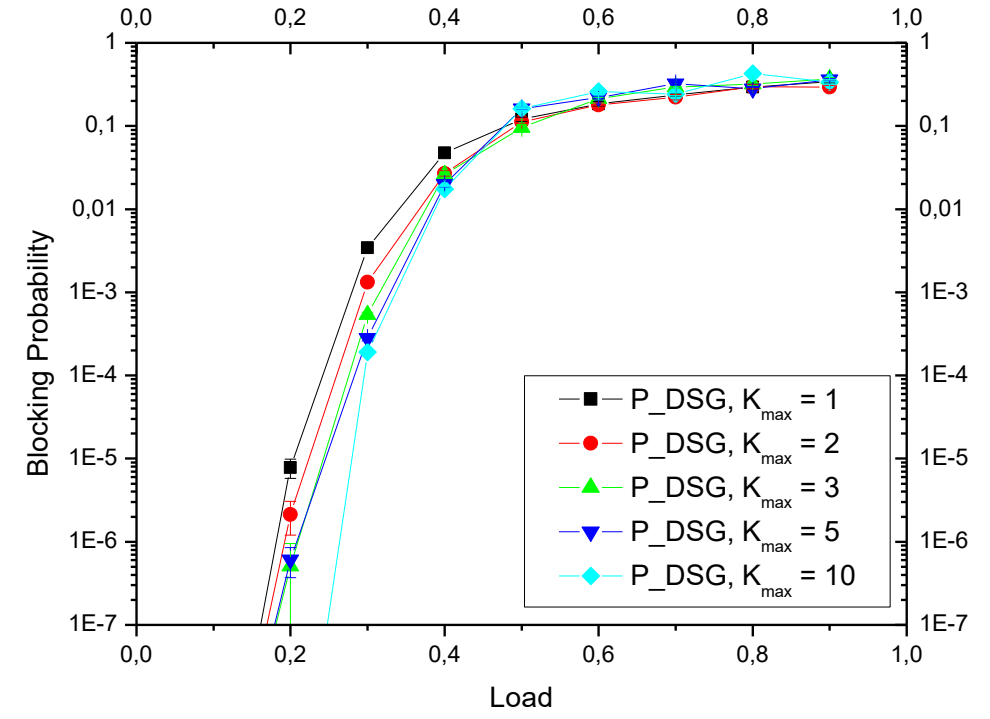


Figura 62. Probabilidad de bloqueo del algoritmo P_DSG-BG para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

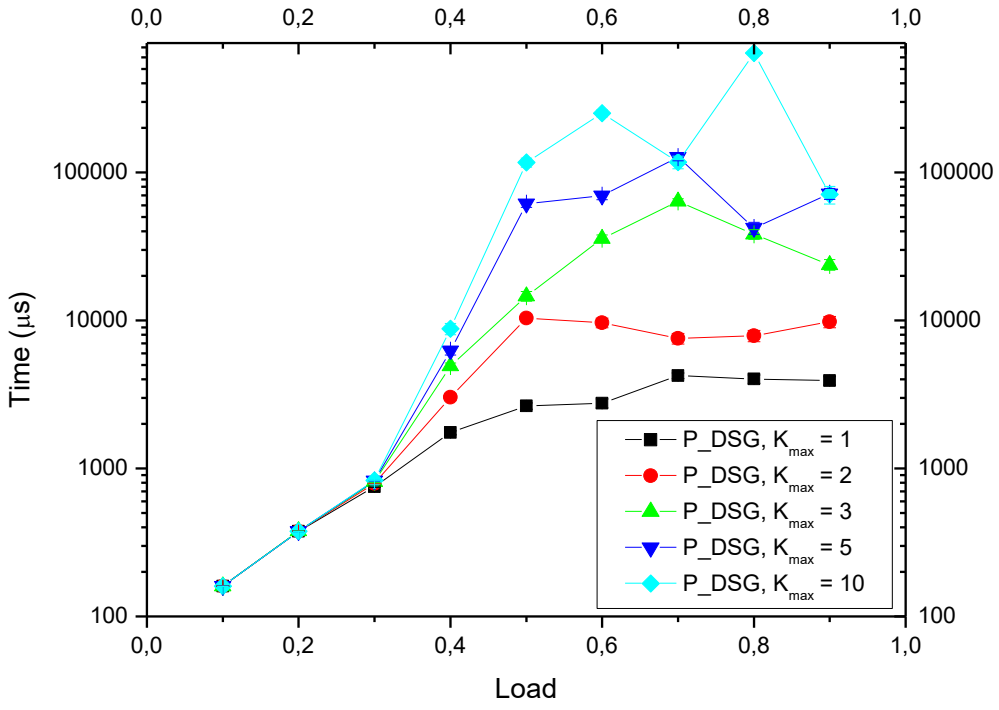


Figura 63. Tiempo de computación del algoritmo P_DSG-BG para varios K_{max}, con ordenación de rutas según shortest-path y barriendo la carga.

Con la Figura 62 y Figura 63 podemos ver que las gráficas son mucho más parecidas a las de otros algoritmos que a las de su propio algoritmo usando *first-fit*, aunque ya lo comprobaremos más adelante en la comparativa general. En el caso sin protección, ocurre algo similar, así que no mostramos las gráficas.

4.7. Comparativa Algoritmos Best-Gap

En esta comparativa, vamos a estudiar los algoritmos ordenados con *shortest-path*, ya que sabemos que con *least-delayed-path* serán un poco menos eficientes. Además, el mejor algoritmo con *shortest-path* será también el mejor con *least-delayed-path*.

Comenzamos mostrando la probabilidad de bloqueo con la protección activada y desactivada en la Figura 64 y Figura 65, respectivamente. Solo mostramos el resultado con 3 caminos porque con el resto obtenemos gráficas muy semejantes.

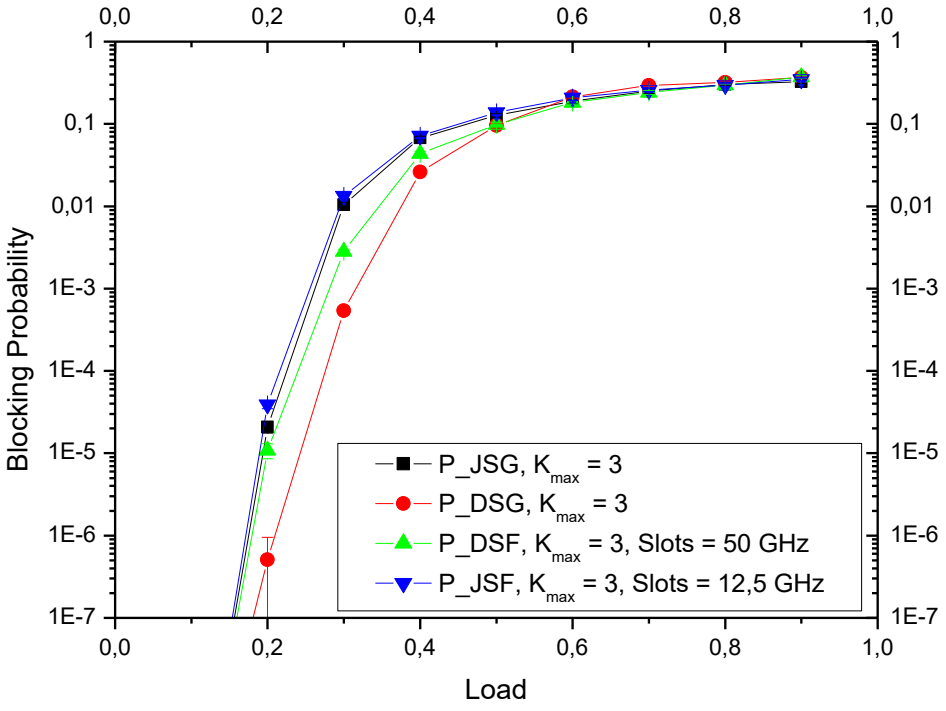


Figura 64. Probabilidad de bloqueo de los algoritmos P_JSG-BG, P_DSG-BG, P_DSF-BG (tamaño de slot de 50 GHz) y P_JSF-BG (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

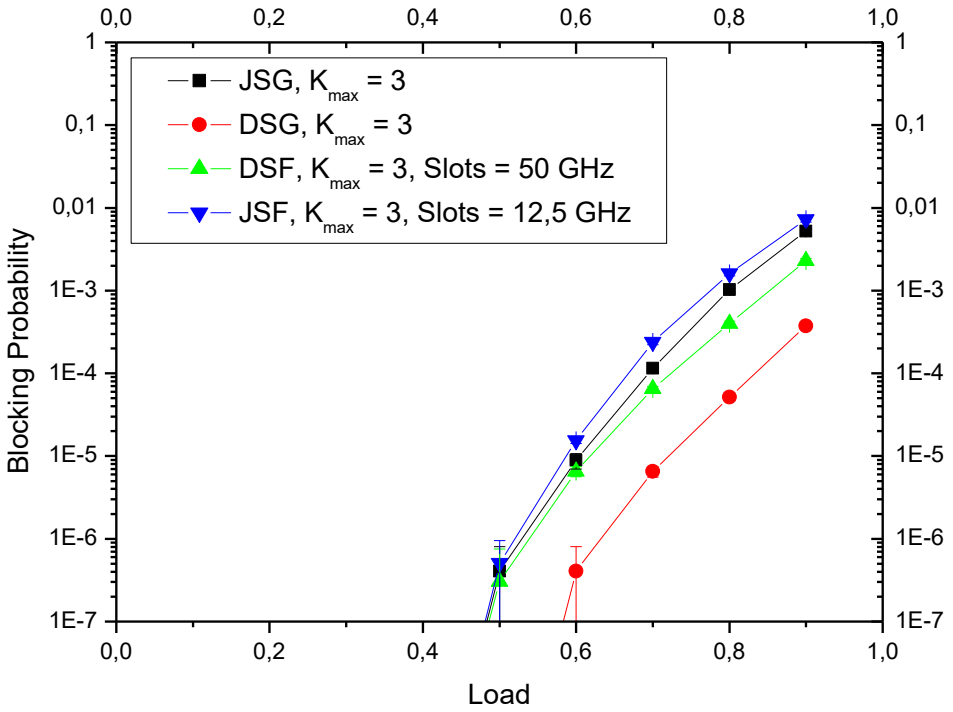


Figura 65. Probabilidad de bloqueo de los algoritmos JSG-BG, DSG-BG, DSF-BG (tamaño de slot de 50 GHz) y JSF-BG (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

Tal y como se puede observar en la Figura 64 y Figura 65, los algoritmos P_DSG-BG y DSG-BG han pasado a ser los más eficientes por bastante diferencia. Esto es debido a que, al elegir el hueco más adaptado, no incurre en un desaprovechamiento del espectro por las bandas de guarda. Además, es el algoritmo más flexible, por lo que todo ello permite que su probabilidad de bloqueo sea muy baja.

Con respecto al tiempo, en la Figura 66 y Figura 67 mostramos una comparación de todos los algoritmos usando rutas de protección y sin usarlas, respectivamente. Al igual que ocurría con la probabilidad de bloqueo, solo lo mostramos para 3 caminos porque para el resto es parecido.

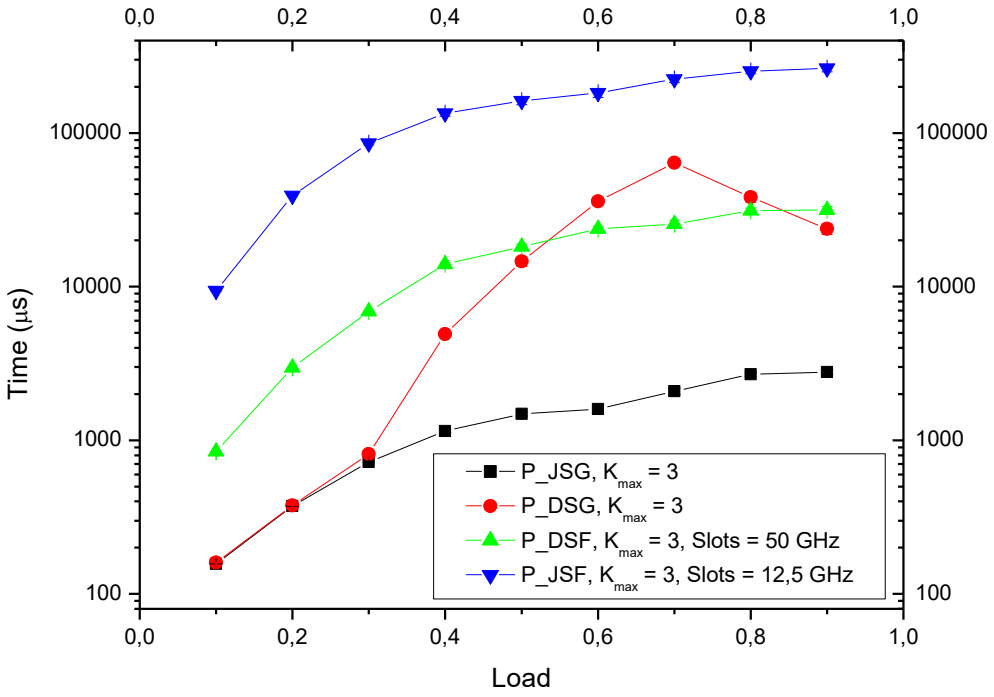


Figura 66. Tiempo de computación de los algoritmos P_JSG-BG, P_DSG-BG, P_DSF-BG (tamaño de slot de 50 GHz) y P_JSF-BG (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

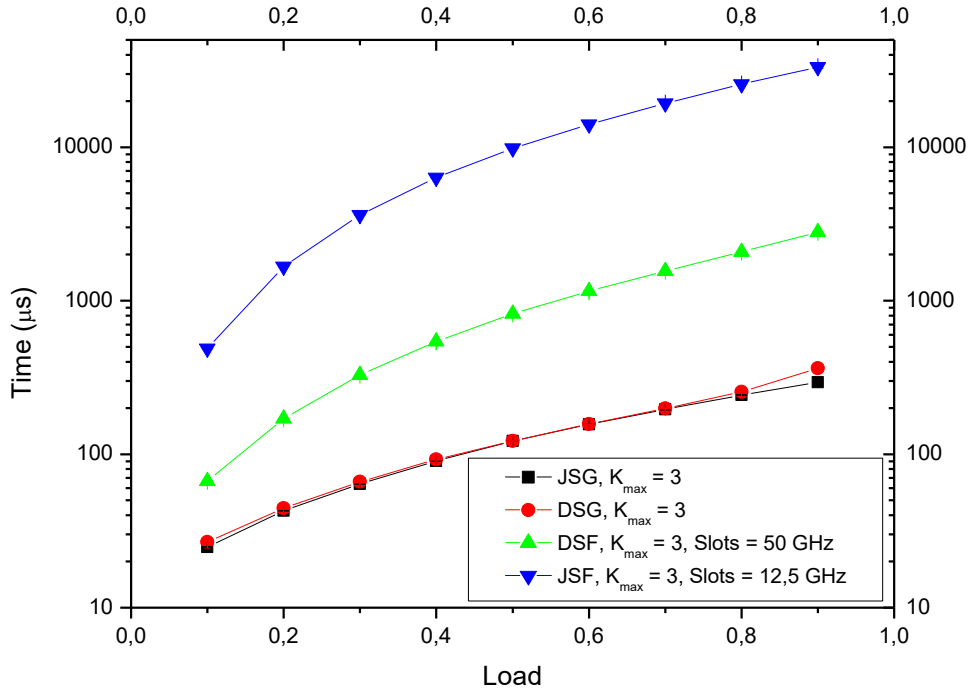


Figura 67. Tiempo de computación de los algoritmos JSG-BG, DSG-BG, DSF-BG (tamaño de slot de 50 GHz) y JSF-BG (tamaño de slot de 12.5 GHz), con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

En la Figura 66 podemos apreciar que el mejor algoritmo es el P_JSG-BG. No obstante, si nos centramos en la zona de trabajo (carga entre 0.2 y 0.3), el tiempo de computación es muy similar al de P_DSG-BG. En la Figura 67 vemos que los algoritmos JSG-BG y DSG-BG tienen una curva prácticamente idéntica. Es por todo ello que podemos afirmar que bajo estas condiciones el mejor algoritmo es el DSG-BG.

4.8. Comparativa General

Para terminar, vamos a comparar *first-fit* y *best-gap* con cada algoritmo, para después comparar los que hemos determinado como más eficientes, esto es, P_JSG-FF y JSG-FF con P_DSG-BG y DSG-BG.

En primer lugar, comparamos los algoritmos JSF, con 10 caminos, rutas ordenadas según *shortest-path* y un tamaño de *slot* de 12.5 GHz, obteniendo la probabilidad de bloqueo y el tiempo medio de computación, tal y como se muestra en la Figura 68 y Figura 69, respectivamente.

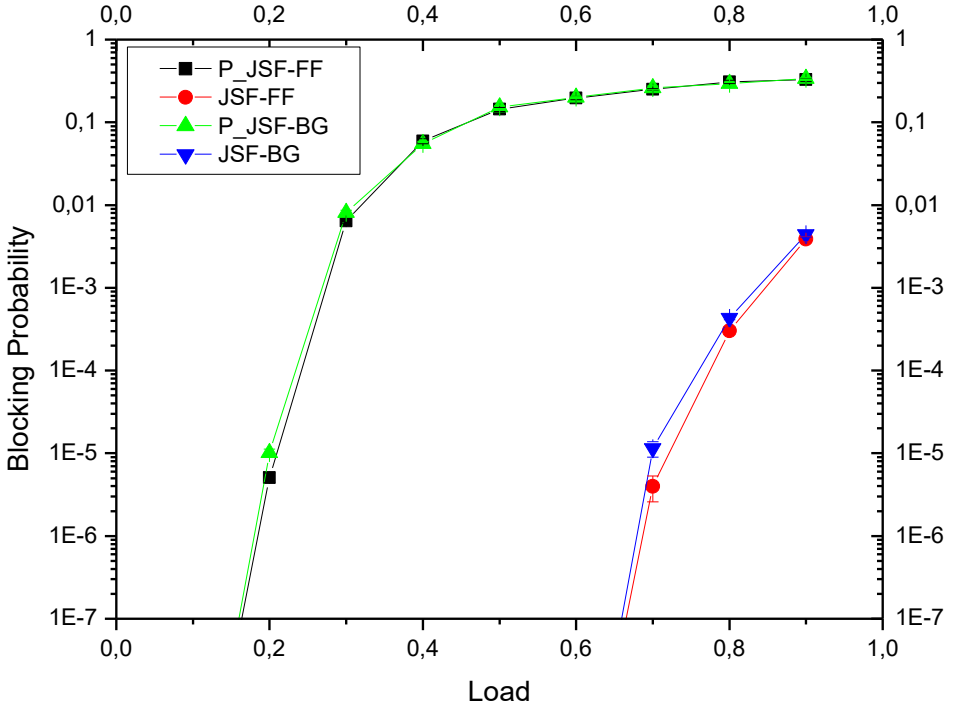


Figura 68. Probabilidad de bloqueo de los algoritmos P_JSF-FF, JSF-FF, P_JSF-BG y JSF-BG, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path, K_{max} = 10 y barriendo la carga.

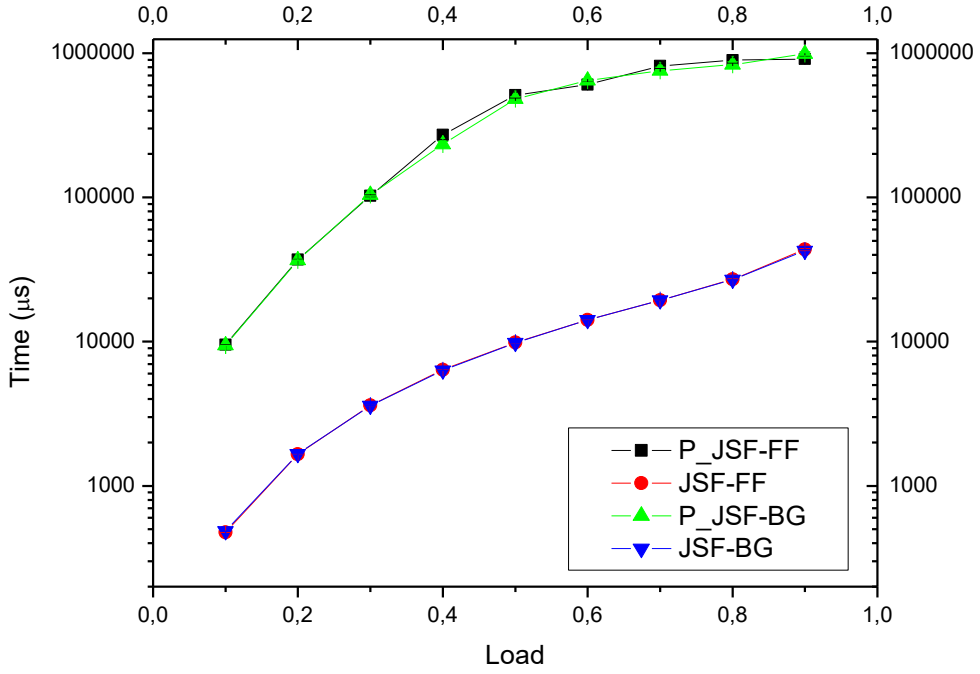


Figura 69. Tiempo de computación de los algoritmos P_JSF-FF, JSF-FF, P_JSF-BG y JSF-BG, con tamaño de slot de 12.5 GHz, ordenación de rutas según shortest-path, K_{max} = 10 y barriendo la carga.

A continuación, en la Figura 70 y Figura 71 mostramos los algoritmos DSF con un número máximo de caminos igual a 5, rutas ordenadas según *shortest-path* y tamaño de *slot* de 50 GHz, obteniendo tanto la probabilidad de bloqueo como el tiempo de computación.

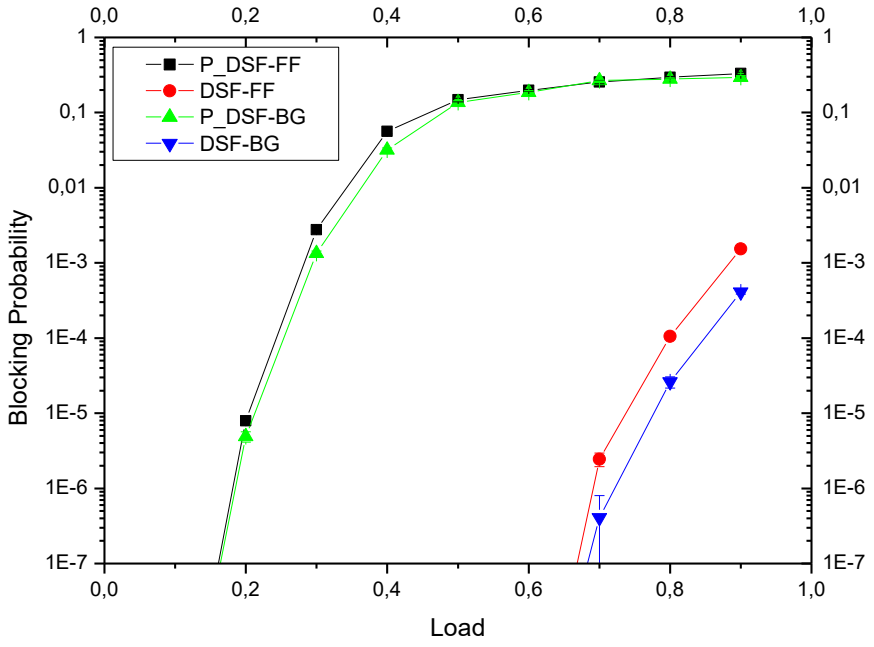


Figura 70. Probabilidad de bloqueo de los algoritmos P_DSF-FF, DSF-FF, P_DSF-BG y DSF-BG, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path, K_{max} = 5 y barriendo la carga.

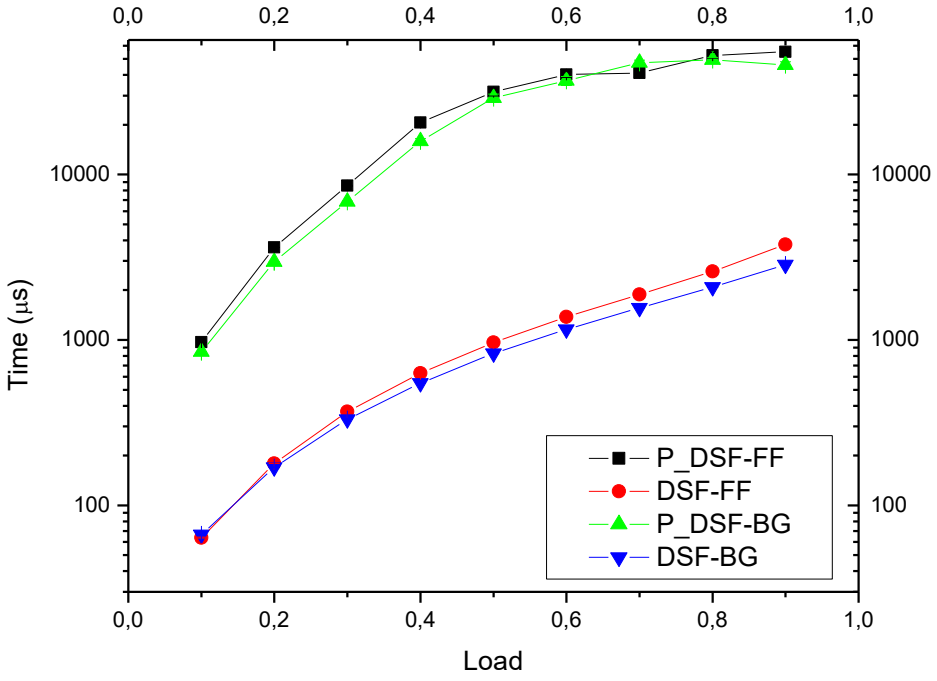


Figura 71. Tiempo de computación de los algoritmos P_DSF-FF, DSF-FF, P_DSF-BG y DSF-BG, con tamaño de slot de 50 GHz, ordenación de rutas según shortest-path, K_{max} = 5 y barriendo la carga.

En la Figura 72 y Figura 73 mostramos la comparativa entre los algoritmos JSG, con 5 caminos y rutas ordenadas según *shortest-path*, obteniendo probabilidad de bloqueo y tiempo de computación, respectivamente.

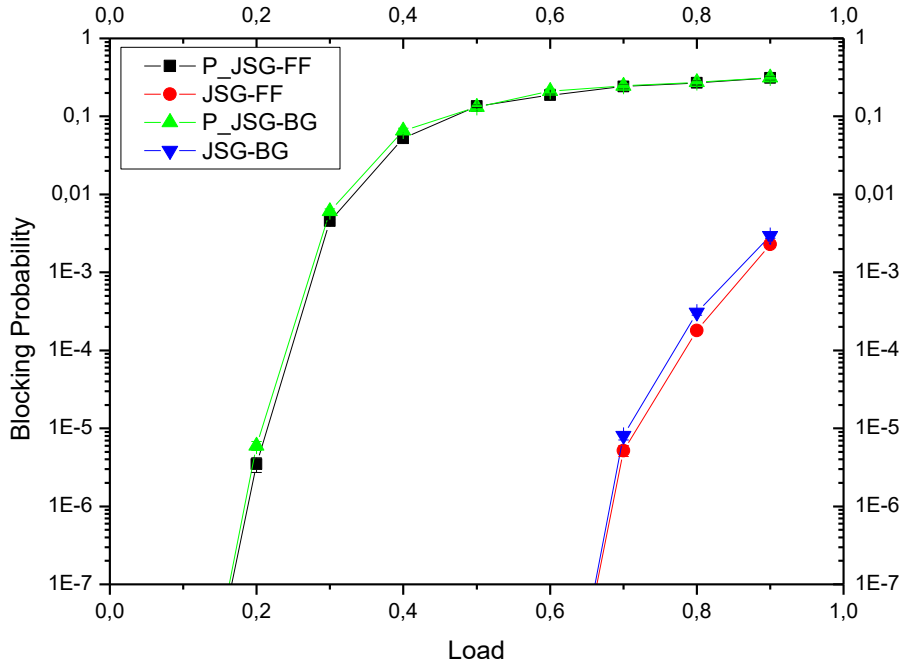


Figura 72. Probabilidad de bloqueo de los algoritmos P_JSG-FF, JSG-FF, P_JSG-BG y JSG-BG, con ordenación de rutas según shortest-path, K_{max} = 5 y barriendo la carga.

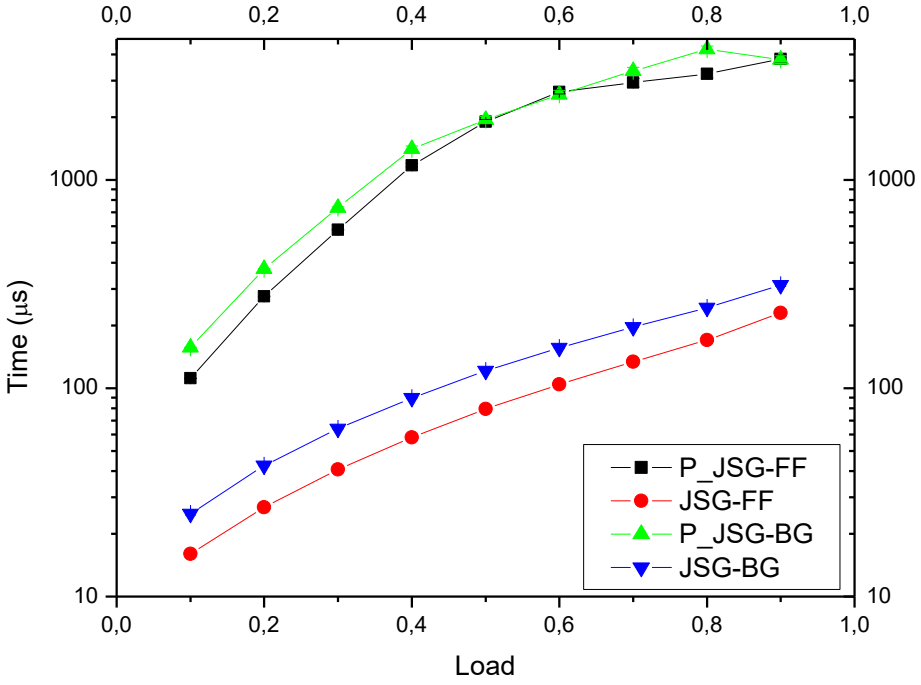


Figura 73. Tiempo de computación de los algoritmos P_JSG-FF, JSG-FF, P_JSG-BG y JSG-BG, con ordenación de rutas según shortest-path, K_{max} = 5 y barriendo la carga.

El último algoritmo que nos queda es DSG, comparativa que realizamos con 3 caminos y rutas ordenadas según *shortest-path*. Se muestra la probabilidad de bloqueo y tiempo de computación en la Figura 74 y Figura 75.

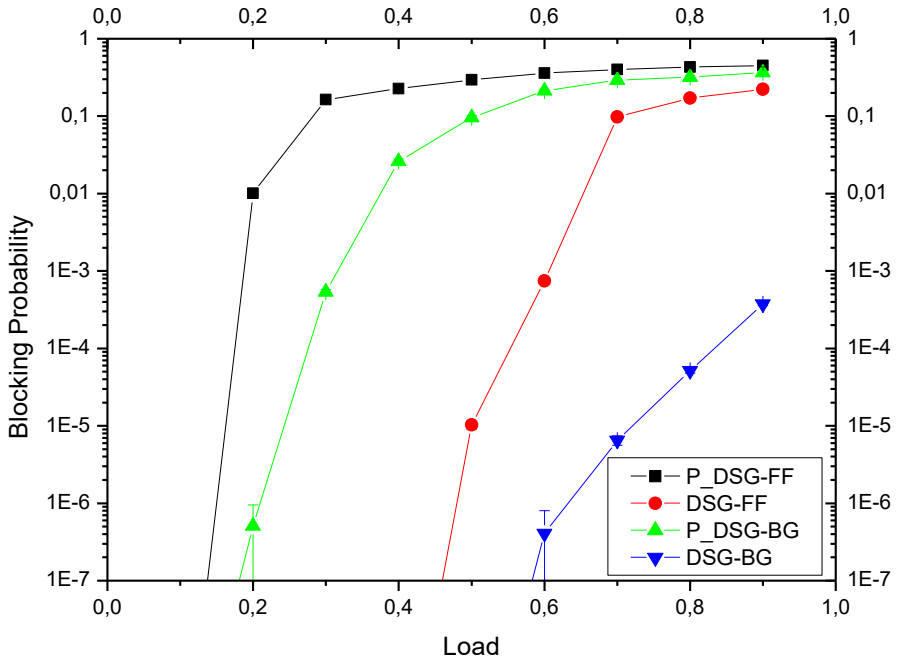


Figura 74. Probabilidad de bloqueo de los algoritmos P_DSG-FF, DSG-FF, P_DSG-BG y DSG-BG, con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

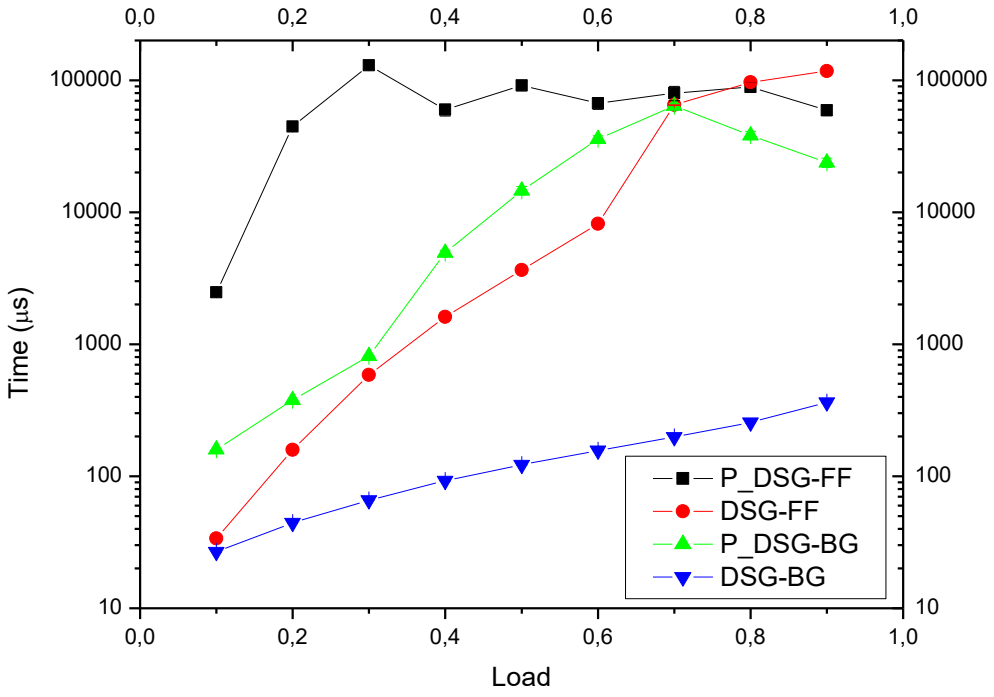


Figura 75. Tiempo de computación de los algoritmos P_DSG-FF, DSG-FF, P_DSG-BG y DSG-BG, con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

En primer lugar, cabe destacar que solo mostramos un número de caminos concreto en cada comparativa. Esto se debe a que las otras gráficas con K_{max} diferentes presentan un aspecto prácticamente idéntico, por lo que no es necesario repetirlas. En cuanto a conclusiones que podemos obtener, vemos que los algoritmos *joint* son ligeramente más eficientes con *first-fit* que con *best-gap*, tanto en probabilidad de bloqueo (Figura 68 y Figura 72) como en tiempo (Figura 69 y Figura 73). Con los algoritmos *disjoint* ocurre lo contrario (Figura 70, Figura 71, Figura 74 y Figura 75), aunque en el algoritmo DSG, como ya previmos, la diferencia es muy significativa, tanto en tiempo como en bloqueo. La explicación a este fenómeno en relación con los *disjoint* se llevó a cabo en el apartado DSG (*best-gap*), y tiene que ver con que se aprovecha la flexibilidad que ofrecen estos algoritmos sin malgastar ancho de banda en bandas de guarda. De hecho, en la Figura 76 se compara el espectro útil del algoritmo P_DSG con *first-fit* y con *best-gap*, donde observamos que el espectro útil en *best-gap* es casi el doble que el de *first-fit*. Esto es debido a que, en *first-fit*, si hay huecos pequeños al principio del espectro, aunque haya un hueco lo suficientemente grande como para albergar la capacidad requerida más adelante, se van asignando los huecos pequeños, lo que conlleva un gasto de ancho de banda causado por las bandas de guarda. Sin embargo, con *best-gap* buscamos el hueco más adaptado y nos ahorramos bandas de guarda. Por otra parte, si analizamos el número medio de rutas por petición en los algoritmos *disjoint*, es decir, número de caminos usados para asignar las capacidades demandadas, en el caso de los algoritmos *best-gap* se obtiene un número algo inferior a los algoritmos *first-fit*, lo que también puede explicar la mayor eficiencia de *best-gap*, incluso en tiempo de computación, a pesar de que recorre más bucles que *first-fit*.

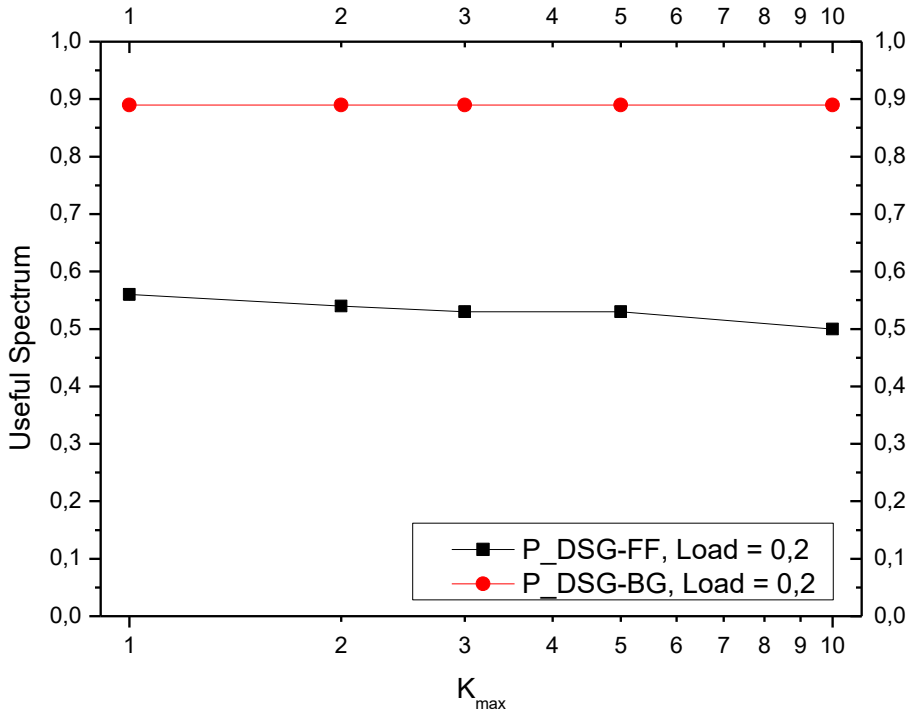


Figura 76. Espectro útil de los algoritmos P_DSG-FF y P_DSG-BG, con ordenación de rutas según shortest-path, carga de 0.2 y barriendo K_{max}.

En relación con los algoritmos *joint*, la mayor eficiencia con respecto al tiempo de computación se puede explicar con el método de programación. Mientras *first-fit* asigna el primer hueco disponible, *best-gap* llama a una función y realiza iteraciones en varios bucles. En relación con la probabilidad de bloqueo, se puede achacar la mayor eficiencia de *first-fit* por la falta de "inteligencia" de *best-gap*, es decir, que a veces el hueco idóneo no es aquel que ofrece la diferencia mínima entre su tamaño y el ancho de banda requerido. Ilustremos esta idea con un ejemplo. Supongamos que el estado del espectro es el de la Figura 77, y que el ancho de banda requerido es una variable aleatoria de media igual a 4 *slots*, bandas de guarda incluidas. Además, queremos asignar un ancho de banda equivalente a 5 *slots*. Es claro que, con *best-gap*, se asignarán los *slots* con flechas coloreadas de azul de encima. Tras asignarlo, quedará un hueco de un *slot*, pero con una media de 4 *slots* por ancho de banda requerido, es poco probable que se llegue a usar ese hueco por sí solo, antes de que se liberen los *slots* a su izquierda o derecha, es decir, se ha fragmentado el espectro. Sin embargo, si asignamos los *slots* de las flechas blancas, quedará un hueco de 4 *slots*, que será más probable que finalmente sea asignado. Esta técnica podríamos decir que se encontraría entre las soluciones para aliviar la fragmentación, explicadas en el capítulo 2.

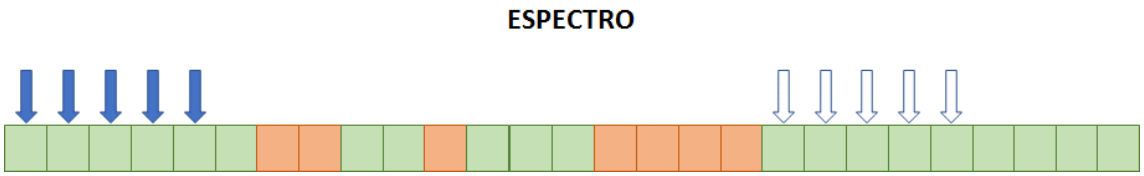


Figura 77. Ejemplo del estado del espectro, y asignación resultante con best-gap (flechas azules) y con best-gap "inteligente" (flechas blancas).

Por último, vamos a comparar los mejores algoritmos, P_JSG-FF y JSG-FF con P_DSG-BG y DSG-BG. Para ello, en la Figura 78 y Figura 79 mostramos la probabilidad de bloqueo y tiempo de computación, respectivamente, para un número máximo de caminos de 3 y rutas ordenadas según *shortest-path*. No mostramos más gráficas para otro número diferente de caminos porque todas son muy parecidas entre sí.

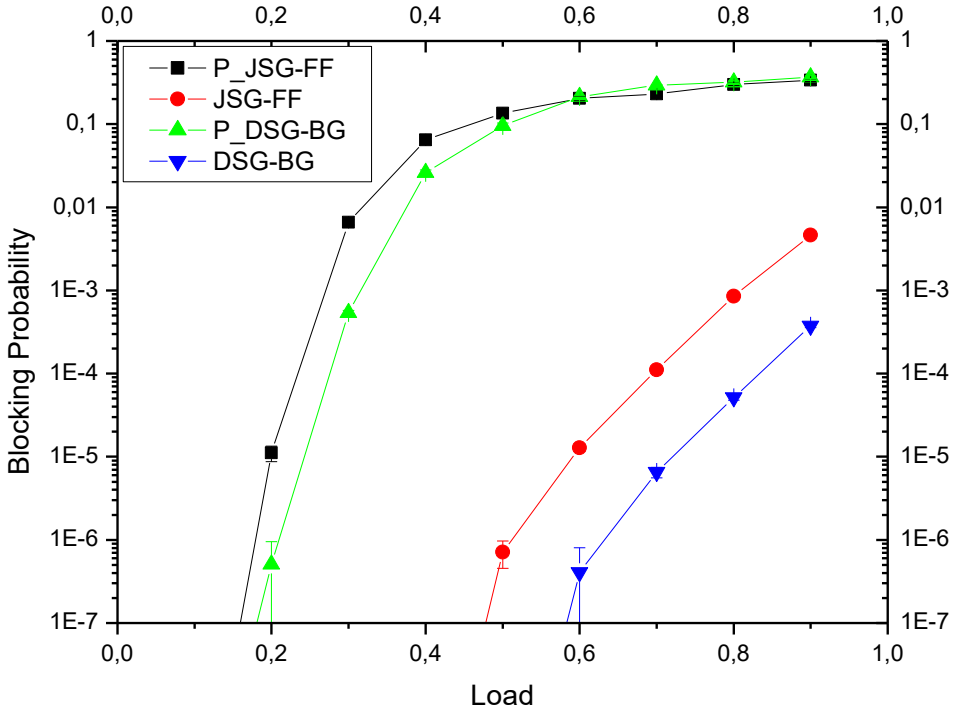


Figura 78. Probabilidad de bloqueo de los algoritmos P_JSG-FF, JSG-FF, P_DSG-BG y DSG-BG, con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

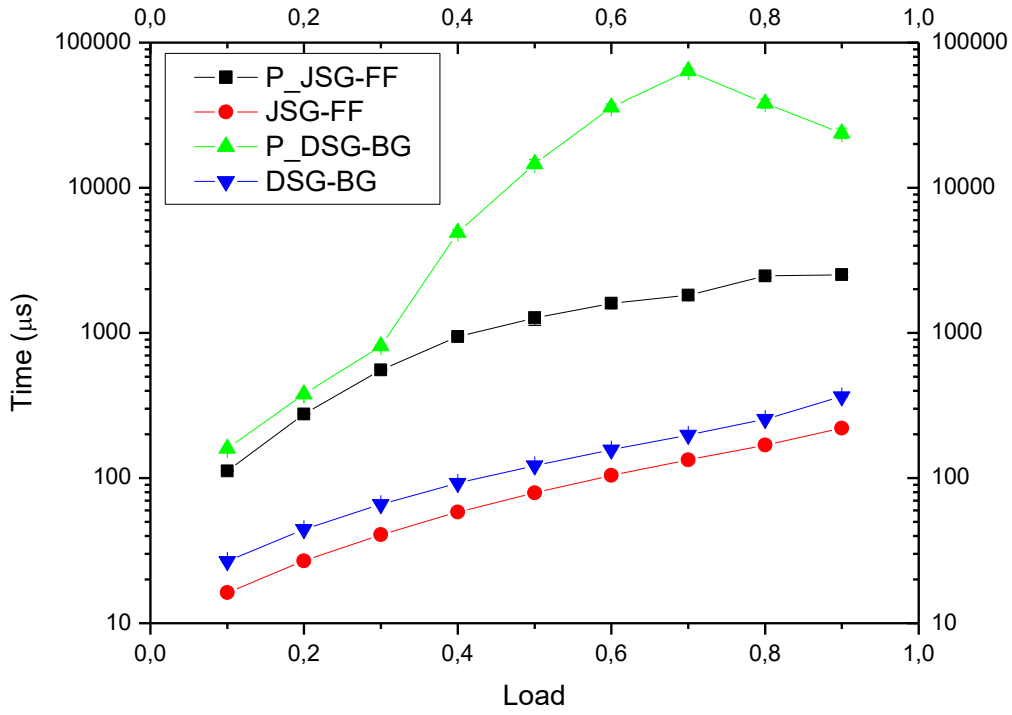


Figura 79. Tiempo de computación de los algoritmos P_JSG-FF, JSG-FF, P_DSG-BG y DSG-BG, con ordenación de rutas según shortest-path, K_{max} = 3 y barriendo la carga.

Analizando la Figura 78 y Figura 79, podemos concluir que los mejores algoritmos son P_DSG-BG y DSG-BG, usando rutas de protección y sin usarlas, respectivamente. Vemos que en probabilidad de bloqueo está muy por encima en cuanto a prestaciones con respecto al otro algoritmo. En cuanto al tiempo de computación, es ligeramente más lento, lo que no es un resultado negativo, sabiendo que los algoritmos P_JSG-FF y JSG-FF son los más rápidos que hemos estudiado. Podemos apreciar en la curva del algoritmo P_DSG-BG de la Figura 79 que, a partir de una carga de 0.4 en adelante, el tiempo pasa a estar muy por encima, pero no supondrá un problema porque la zona de trabajo es hasta una carga de 0.3, donde el tiempo se mantiene muy similar al del algoritmo P_JSG-FF.

Capítulo 5. Conclusiones

En este TFG hemos visto la importancia de las redes ópticas elásticas, que se están perfilando como la solución ideal ante la creciente demanda de servicios que requieren gran cantidad de ancho de banda. Gracias a estas redes se consigue un uso del espectro eficiente, ya que adaptan sus recursos a las peticiones de conexión. También hemos analizado lo relevante que es contar con técnicas de supervivencia en dichas redes, principalmente por la gran cantidad de datos que se transmiten y, en consecuencia, por las pérdidas que acarrearía no transmitir cierta información crítica. Además, se han expuesto las líneas de investigación actuales en relación a estas redes, prestando especial atención a la supervivencia y a las nuevas tecnologías de control, como SDN, y se han indicado artículos publicados similares al objetivo de este trabajo.

A lo largo de este Trabajo Fin de Grado, hemos propuesto 16 algoritmos de asignación de recursos, tal y como se muestra en la Tabla 1. En ellos, hemos tenido en cuenta los siguientes parámetros:

- Búsqueda de un ancho de banda compacto o disgregado
- Uso de rejilla de canales o uso flexible del espectro
- Búsqueda del hueco más adaptado a la demanda
- Inclusión de técnicas de protección

Tras realizar una descripción detallada de cada algoritmo, hemos llevado a cabo simulaciones en la herramienta OMNeT++ para determinar cuál de ellos ofrece mejores prestaciones, tanto en probabilidad de bloqueo como en tiempo de computación. La probabilidad de bloqueo es muy importante por el hecho de que buscamos que se atiendan el máximo número de peticiones posibles. Respecto al tiempo de computación, puesto que utilizamos un mecanismo dinámico de asignación de recursos, es también fundamental obtener tiempos pequeños. Entre las conclusiones obtenidas, las más relevantes son las siguientes:

- Los algoritmos *first-fit* son mejores que los *best-gap* cuando nos encontramos en el caso *joint*, mientras que los algoritmos *best-gap* son mejores cuando los combinamos con el caso *disjoint*. Esto se debe principalmente a que la flexibilidad que ofrecen los algoritmos *disjoint* se aprovecha mejor con *best-gap*, ya que no malgastamos ancho de banda debido a bandas de guarda innecesarias. En relación con el caso *joint*, hemos achacado la mayor eficiencia de *first-fit* a la falta de "inteligencia" de *best-gap*. En [15], se propone un algoritmo llamado *exact-fit* que, como ya hemos comentado, no es aplicable al caso *gridless*. No obstante, el razonamiento que detalla puede ser útil, y lo podríamos aplicar a *best-gap*, de forma que cuando no se encuentre un hueco lo suficientemente adaptado, se elija el hueco más grande existente. De esta forma, creemos que se obtendrían mejores resultados.
- El uso de un esquema de protección es claramente más ineficiente que no usarlo. Esto es evidente, porque reservamos más del doble de recursos,

obteniendo mayores probabilidades de bloqueo y tiempos de computación. No obstante, como ya hemos comentado, la supervivencia en las redes ópticas elásticas se ha convertido en un asunto de vital importancia. Existe por tanto un compromiso entre el grado de supervivencia de la red y la eficiencia. Ante esta encrucijada, creemos que el uso de protección compartida ofrecería buenas prestaciones, tanto en el grado de supervivencia como en la eficiencia, por lo que sería una buena solución a este problema.

- El uso de rejilla de canales incurre en desaprovechamientos del espectro, debido a que hay que rellenar los *slots* al completo. Por tanto, usando espectros *gridless* obtenemos una eficiencia mayor. Sin embargo, es importante prestar atención al método de asignación, pues el algoritmo *Disjoint Spectrum Gridless* (*First-Fit*) es muy ineficiente, dado que se asignan multitud de huecos pequeños, desaprovechando gran parte del espectro con bandas de guarda.
- Con la asignación *first-fit*, el algoritmo más eficiente es *Joint Spectrum Gridless*, mientras que con *best-gap* obtenemos claramente mejores prestaciones con *Disjoint Spectrum Gridless*. Comparando ambos algoritmos podemos determinar que DSG-BG es muy superior al resto en probabilidad de bloqueo y algo inferior a JSG-FF en tiempo de computación, ambas conclusiones obtenidas con y sin el uso de esquemas de protección. Por consiguiente, concluimos que si usáramos DSG-BG o P_DSG-BG obtendríamos la mayor eficiencia entre los algoritmos vistos.

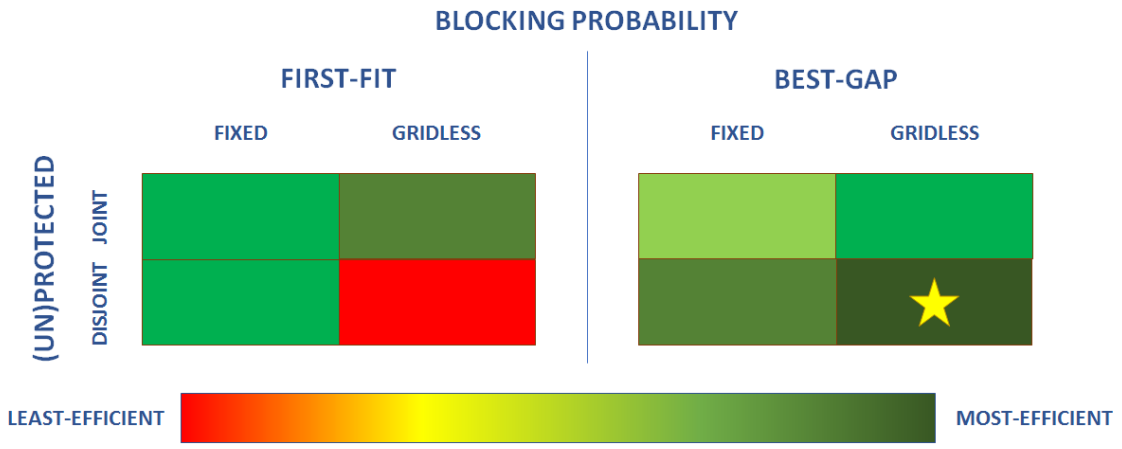


Figura 80. Resumen de la eficiencia de los algoritmos en función de la probabilidad de bloqueo.

En la Figura 80 se muestra un resumen gráfico de las conclusiones obtenidas acerca de la eficiencia de los algoritmos en función de la probabilidad de bloqueo. En primer lugar, podemos ver que no hay diferencia entre usar o no esquemas de protección a la hora de ordenar los algoritmos según su eficiencia. En segundo lugar, apreciamos que el mejor algoritmo es *Disjoint Spectrum Gridless – Best-Gap* (estrella amarilla). Además, mostramos la eficiencia de todos ellos a través de una escala de colores, de modo que los algoritmos más eficientes están asociados a un verde más oscuro, mientras que los menos eficientes se asocian a colores verdes claros, o rojo para el caso de ineficiencia.

Como líneas futuras, se propone dotar de más "inteligencia" a los algoritmos *best-gap*, de forma que si el hueco más adaptado es lo suficiente mayor como para fragmentar el espectro, se asigne el ancho de banda al hueco más grande existente. Además, se podría desarrollar un método de protección compartida, ya que el esquema de protección implementado en este trabajo es muy útil con aplicaciones críticas, pero con el resto resulta ineficiente. Con este esquema además del de protección compartida, se podría elegir qué protección otorgar a cada conexión en función de su prioridad. Por último, podría desarrollarse un algoritmo que resolviera el problema RSA de forma conjunta, es decir, *joint RSA*, y elegir aquellos algoritmos que minimizaran el tiempo de computación. Tras ello, se podrían comparar dichos resultados con los obtenidos en este trabajo.

Referencias

[1] R.-J. Essiambre and R.W. Tkach, "Capacity Trends and Limits of Optical Communication Networks", Proceedings of the IEEE, vol. 100, no. 5, pp. 1035-1055, May 2012.

[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] 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.

[7] D. Zhou and S. Subramaniam, "Survivability in Optical Networks", IEEE Network, vol. 14, no. 6, pp. 16-23, November/December 2000.

[8] S. Kim and S.S. Lumetta, "Multiple Failure Survivability in WDM Mesh Networks", Coordinated Science Laboratory, University of Illionis at Urbana-Champaign, May 2006.

[9] P.N. Ji, "Softare Defined Optical Network", The 2012 11th International Conference on Optical Communications and Networks (ICOCN), 2012.

[10] A.S. Thyagaturu, A. Mercian, M.P. McGarry, M. Reisslein, and W. Kellerer, "Software Defined Optical Networks (SDONs): A comprehensive Survey", IEEE Communications Surveys & Tutorials, vol. 18, no. 4, pp. 2738-2786, Fourth Quarter 2016.

[11] OMNeT++ Discrete Event Simulator. https://omnetpp.org/.

[12] 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.

[13] 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.

[14] 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.

[15] A. Rosa, C. Cavdar, S. Carvalho, J. Costa, and L. Wosinska, "Spectrum Allocation Policy Modeling for Elastic Optical Networks", High Capacity Optical Networks and Emerging/Enabling Technologies, pp. 242-246, 2012.

[16] M. Klinkowski, "An Evolutionary Algorithm Approach for Dedicated Path Protection Problem in Elastic Optical Networks", Cybernetics and Systems: An International Journal, vol. 44, no. 6-7, pp. 589-605, August 2013.

[17] O. Astrachan, "Bubble Sort: an Archaeological Algorithmic Analysis", Computer Science Department, Duke University, March 2003.

[18] ITU-T, *Recommendation G.694.1 : Spectral grids for WDM applications: DWDM frequency grid*, February 2012.

[19] M.S. Kumar and P.S. Kumar, "Lightpath Setup Time Optimization in Wavelength Routed All-Optical Networks", Computer Communications, vol. 24, no. 10, pp. 984-995, May 2001.