MENÚ SALIR

Universidad de Extremadura

Departamento de Tecnología de los Computadores y de las Comunicaciones

Tesis Doctoral

Metaheurísticas y computación paralela para el problema de la planificación de frecuencias en redes reales de telecomunicaciones

---

Autor
José Manuel Chaves González

Director
Dr. Miguel Ángel Vega Rodríguez

Tesis presentada para optar al Grado de Doctor Ingeniero en Informática
Escuela Politécnica, Cáceres, España
Abril de 2011

Edita: Universidad de Extremadura
Servicio de Publicaciones

Caldereros 2. Planta 3ª
Cáceres 10071
Correo e.: publicac@unex.es
http://www.unex.es/publicaciones

Resumen

El diseño de técnicas y algoritmos eficientes que resuelvan adecuadamente problemas complejos de optimización es uno de los campos dentro de la investigación en Informática con mayor repercusión en la actualidad. De hecho, debido a que la sociedad actual demanda continuamente más y mayores retos en el campo de la ingeniería, dar una solución adecuada y eficiente a este tipo de problemas cobra cada vez mayor importancia tanto para la industria como para la comunidad científica. Además, es bien sabido que tanto los requisitos temporales como computacionales son un factor crítico a tener en cuenta cuando se trata de resolver problemas de grandes proporciones cuya solución debe ser aplicable en un supuesto real. En estos casos, el uso de heurísticas y metaheurísticas en combinación con estrategias paralelas se presenta como la mejor alternativa de resolución, ya que por una parte, las citadas estrategias no exactas proporcionan una solución de alta calidad en un tiempo razonable, mientras que por otro, es bien sabido que tanto la eficiencia de dichos algoritmos heurísticos como los resultados que éstos obtienen mejoran de manera significativa cuando se aplican estrategias de paralelismo. Éste es el contexto global en el que se incardina esta tesis doctoral.

Por tanto, se han estudiado, diseñado y desarrollado un conjunto heterogéneo de metaheurísticas para resolver un importante problema de optimización dentro del dominio de las Telecomunicaciones. Sin embargo, el uso de algoritmos metaheurísticos conlleva un alto coste computacional, ya que por una parte, dichos métodos incluyen una gran cantidad de parámetros que es necesario ajustar mediante numerosos experimentos, mientras que por otra, ciertas partes de dichos algoritmos requieren mucha capacidad de cómputo. Por consiguiente, a la hora de abordar problemas reales de grandes dimensiones, como es el caso del problema manejado en esta tesis, tanto los requisitos técnicos como temporales requeridos por estas técnicas son muy exigentes. Así

pues, se hace imprescindible que el uso de las técnicas metaheurísticas esté combinado con estrategias paralelas. De esta forma, se mejora tanto la productividad como el rendimiento de las metaheurísticas, ya que de una parte el paralelismo permite que el ajuste paramétrico de las metaheurísticas se agilice considerablemente, haciendo que este proceso sea más preciso y exhaustivo, mientras que de otra se mejora notablemente la calidad de los resultados en tiempos que serían más elevados si los algoritmos se ejecutaran de manera secuencial. Las estrategias paralelas para mejorar productividad y rendimiento son diferentes, ya que en el primer caso se utilizan paradigmas diseñados para aplicar computación de alta productividad y en el otro se utilizan modelos aplicados a computación de alto rendimiento. En esta tesis se han utilizado ambos modelos, haciendo uso de estrategias paralelas diseñadas para ser ejecutadas tanto en entornos clúster como en entornos grid.

En concreto, el problema de optimización con el que se ha trabajado en esta tesis es el de la asignación automática de frecuencias (FAP, del inglés *Frequency Assigment Problem*), y fue elegido, entre otras razones, tanto por su complejidad (se trata de un problema NP-Completo) como por su relevancia para la industria del sector, ya que es bien conocido que el mundo actual en que vivimos está dominado por las comunicaciones, lo cual provoca que los recursos necesarios para llevar a cabo este proceso sean muy limitados. Este hecho hace que las compañías que ofrecen servicios móviles deban gestionar los citados recursos de manera eficiente para sacar el máximo rendimiento de ellos manteniendo unos niveles altos en la calidad de los servicios que ofrecen.

El problema de optimización que surge con el FAP se debe al reducido rango de frecuencias disponible para cada red de telefonía. Debido a este hecho, no es posible asignar a cada comunicación que se establece en la red una frecuencia única, sino que los operadores deben repetir las frecuencias de que disponen múltiples veces para cubrir todas las comunicaciones que se producen entre los usuarios de la red. Sin embargo, este solapamiento de frecuencias causa interferencias que dificultan, e incluso pueden llegar a anular, dichas comunicaciones, por lo que se hace necesario realizar una planificación de frecuencias eficiente con la que se consiga maximizar el número de comunicaciones que se realizan en la red manteniendo a la vez unos mínimos aceptables en la calidad de los servicios ofrecidos.

Tras hacer una amplia revisión de los algoritmos heurísticos y metaheurísticos que parecían ajustarse mejor a los requisitos del problema descrito anteriormente, se terminó por desarrollar y ajustar un conjunto de siete metaheurísticas con el que se trató de resolver el problema de la asignación de frecuencias desde varios enfoques. El primero que se implementó fue PBIL (aprendizaje incremental basado en población), el cual se aplicó primero a una versión *benchmark* del FAP (las instancias Philadelphia), para después ser adaptado a la versión real del problema que ha centrado la investigación del autor de esta tesis. Después se implementó el algoritmo SS (búsqueda dispersa), que ha sido una de las aproximaciones que mejores resultados ha obtenido en la resolución del problema. No se quiso dejar fuera del estudio al clásico algoritmo genético (GA), que tras un exhaustivo ajuste demostró proporcionar planificaciones de frecuencia de muy alta calidad. Finalmente, se incluyó un algoritmo basado en inteligencia colectiva, el ABC (algoritmo basado en colonia de abejas, de reciente aparición), que también ofreció muy buenos resultados. Todos los algoritmos mencionados anteriormente son metaheurísticas basadas en población, ya que utilizan una población de soluciones en su búsqueda del óptimo global, pero no se quisieron dejar fuera del estudio estrategias basadas en trayectoria, ya que presentan características que también resultan interesantes. Así, se desarrollaron los algoritmos ILS (búsqueda local iterativa), VNS (búsqueda de entorno variable) y GRASP (procedimiento de búsqueda adaptativa, avariciosa y aleatoria), con el que además se desarrolló un modelo paralelo maestro-esclavo utilizando una infraestructura grid real.

Por tanto, se realizaron múltiples pruebas y experimentos con cada metaheurística, aplicando en algunos casos paradigmas basados en paralelismo. Sin embargo, el estudio presentado en esta tesis concluyó con el diseño e implementación de una novedosa estrategia paralela que hacía uso de todas las metaheurísticas desarrolladas. En concreto, se desarrolló una hiperheurística paralela basada en el heterogéneo conjunto de metaheurísticas que se había desarrollado, y que dio como resultados planificaciones de frecuencia de muy alta calidad que solucionaban el problema de optimización abordado. De hecho, entre las principales contribuciones aportadas por esta tesis se encuentran, por un lado, el desarrollo y evaluación de metaheurísticas que no habían sido aplicadas antes en la resolución del problema FAP, obteniéndose además

resultados de muy buena calidad en todas ellas, y por otro, el diseño e implementación de una eficiente estrategia paralela (la hiperheurística basada en metaheurísticas) con la que se ha conseguido mejorar, hasta donde nosotros conocemos, cualquier resultado (tanto en tiempo como en calidad) publicado hasta la fecha en la resolución del problema de la asignación automática de frecuencia en redes reales de telecomunicaciones.

Abstract

The design of efficient algorithms which can be applied to the resolution of complex optimization problems is one of the main goals of the research in Computer Science. Due to the new engineering challenges that our society demands in our days, these problems are very relevant for both the industry and the scientific community. Moreover, it is well known that time and computational requirements are important parameters which must be specially considered in case of realistic-sized complex optimization problems. In those real-world cases, heuristic and metaheuristic methods along with parallel strategies are widely accepted as one of the best resolution strategies, because on the one hand, these techniques provide high quality solutions in a reasonable period of time, and on the other hand, the results they can provide and their efficiency improve significantly when they are combined along with parallel strategies. This is the context for this thesis.

Therefore, we have studied, designed and developed a heterogeneous set of metaheuristics to resolve an important optimization problem within the Telecommunication domain. However, the use of metaheuristics involves a high computation cost, because on the one hand these strategies require a large number of experiments to adjust the diverse number of parameters they include, and on the other hand some significant parts of those algorithms require a high computational power to be executed. Thus, it is mandatory that metaheuristics are combined with parallel strategies when they tackle large real-world optimization problems. Both, the throughput and the performance of metaheuristics will be significantly improved using parallel computing because the parametric adjustment of those techniques will be performed faster (in this case we refer to high throughput computing –HTC), but once the algorithms are adjusted, the results obtained will be also of more quality if they are implemented using a parallel model (high performance computing –HTC). In this

thesis both parallel approaches have been used using cluster and grid environments.

Specifically, the problem addressed here is the Frequency Assignment Problem (FAP), which is an important optimization problem taken from the Telecommunication domain. This problem was chosen for a double reason: first, it is a very complex problem (in fact, it is NP-hard) and second, it is very relevant for the current industry, because it is well known that communications are essential in the world in which we live today and the efficient management of the limited available resources used in this process is absolutely mandatory for mobile operators. Therefore, the optimization problem which arises from the FAP is that due to the narrow spectrum of frequencies which are available for each network, frequencies have to be repeated so that the operators can cover all communications produced between any pair of cell phones subscribed to that network. This frequency overlapping causes interferences which make difficult, or even impossible, a communication of quality. Therefore, it is necessary that each phone network has an efficient frequency plan which is able to obtain the maximum coverage but keeping a minimum level of quality in the service offered.

With the aim of achieving this goal, we studied a wide set of heuristic and metaheuristic approaches which can be applied to the FAP. After this study, we developed a set of seven metaheuristics with the support of parallel strategies. First, we used PBIL (population based incremental learning) to resolve a FAP benchmark (the Philadelphia instances), and after that we adjusted the algorithm to the real-world version of the problem which has focused the research of the author of this thesis. Second, we developed the SS (Scatter Search) metaheuristic, which was one of the approaches which obtained better results in the resolution of the problem. We did not want to remove from our study the classical GA (genetic algorithm), which provided high quality frequency plans after a thoroughly parameter adjustment. We also included a swarm intelligence algorithm, the ABC (artificial bee colony algorithm, recently appeared), which obtained as well very good results. All the metaheuristics mentioned previously are population based algorithms, because they use a population of solutions in the search of an optimal solution. But we also included in our study trajectory based metaheuristics, because they present interesting

features, such as a very fast evolution. Thus, we developed and fitted to represent this kind of algorithms ILS (iterated local search), VNS (variable neighbourhood search) and GRASP (greedy randomized adaptive search procedure) metaheuristics.

We performed a large number of experiments with each metaheuristic, applying parallel models in some cases, and after a complete study, we designed a newly parallel strategy (a hyperheuristic based on metaheuristics) with the heterogeneous set of metaheuristics mentioned previously. This approach obtains high quality solutions for the optimization problem solved in our study. In fact, among the main contributions of this thesis are the development and evaluation of some metaheuristics which have not been applied before to the resolution of the FAP, and the design and implementation of a novel parallel strategy (a parallel hyper-heuristic) which, to our best knowledge, improves the results (both in time and in quality) obtained by any other approach published so far applied to the resolution of the optimization problem tackled in this thesis.

Agradecimientos

La realización de esta tesis no habría sido posible sin la ayuda de muchas personas e instituciones que han colaborado con el autor de este documento de muy diversas maneras. Para empezar, quiero agradecer su apoyo y dedicación a mi director, Miguel Ángel, sin cuya ayuda, motivación y dirección esta tesis nunca habría visto la luz. Su entrega ha ido más allá de la mera dirección técnica o científica, dedicando incontable tiempo, esfuerzo y paciencia a apoyarme y guiarme cuando más lo he necesitado.

Por otro lado, merecen especial mención la Junta de Extremadura, el Ministerio de Ciencia e Innovación y el FEDER, ya que sin el soporte económico proporcionado por la beca FPI PRE06003 o la financiación conseguida por los proyectos OPLINK (TIN2005-08818-C04-03) y MSTAR (TIN2008-06491-C04-04) la investigación llevada a cabo durante estos años no habría sido posible.

Pero la ayuda conseguida mediante los citados proyectos no ha sido sólo económica, ya que gracias a ellos se realizaron estrechas colaboraciones que fueron de gran importancia para la realización del trabajo que aquí se presenta. Así, me gustaría agradecer de manera especial la generosa contribución realizada por todos los miembros del consorcio que han participado de alguna manera con la investigación llevada a cabo en esta tesis. En particular, merece especial mención la ayuda y el soporte científico-técnico proporcionados por el Dr. Francisco Luna, de la Universidad de Málaga, cuyo conocimiento del problema y experiencia han resultado vitales para la investigación realizada. También es destacable la colaboración de Carlos Segura, de la Universidad de La Laguna, que desarrolló una búsqueda local que se comporta de manera extraordinaria en la resolución del problema abordado en esta tesis.

Pero volviendo de nuevo al ámbito más cercano al autor, me gustaría también agradecer a Juan Manuel y Juan Antonio los muchos ánimos, el apoyo y los sabios consejos ofrecidos durante todos estos años. Intento tratar a la gente nueva que llega al grupo tan bien como me tratasteis a mi cuando comencé a trabajar con vosotros. Me siento realmente afortunado de pertenecer a un grupo con tan alta calidad humana. Además, no puedo ni debo olvidar a mis compañeros de laboratorio, y amigos, José María, Álvaro, David y Mateo, con los que he pasado muy buenos ratos en estos años y que siempre han estado ahí para animarme y ayudarme con los contratiempos cotidianos que tiene la labor de investigador. ¡Muchas gracias a todos!

A mi familia y amigos les debo su apoyo incondicional y todo el cariño que me dan. Merecen especial mención en este punto mi madre y mi hermano, a los que quiero de manera infinita. Quiero expresar aquí que si he llegado hasta donde estoy ahora es en gran parte gracias a mi madre. Sin duda ella es la persona que más me ha apoyado y que más se ha sacrificado tanto por mí como por mi hermano. He de decir en este punto que si hoy día tengo algún valor moral o algún atributo como persona que merezca la pena es en gran parte gracias a ella, por lo que quiero expresar en estas líneas mi más afectuoso reconocimiento y gratitud.

Y qué decir de mis mejores colegas, José Carlos y Noé, con los que tan buenos momentos he compartido y a los que tanto les debo. Ellos saben la altísima estima que les tengo, por lo que no puedo hacer aquí otra cosa que agradecerles su apoyo y amistad sincera.

Finalmente, quiero agradecerle su comprensión, aliento y cariño a la que es mi compañera, mi amor y mi vida; a Raquel, muchas gracias por el regalo que me haces cada día.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y motivación

El diseño de técnicas y algoritmos eficientes que resuelvan adecuadamente problemas complejos de optimización es uno de los campos dentro de la investigación en Informática con mayor repercusión en la actualidad. De hecho, debido a que la sociedad actual demanda continuamente más y mayores retos en el campo de la ingeniería, dar una solución adecuada y eficiente a este tipo de problemas cobra cada vez mayor importancia tanto para la industria como para la comunidad científica. Además, es bien sabido que tanto los requisitos temporales como computacionales son un factor crítico a tener en cuenta cuando se trata de resolver problemas de grandes proporciones cuya solución debe ser aplicable en un supuesto real. En estos casos, el uso de heurísticas y metaheurísticas en combinación con estrategias paralelas se presenta como la mejor alternativa de resolución, ya que por una parte, las citadas estrategias no exactas proporcionan una solución de alta calidad en un tiempo razonable, mientras que por otro, es bien sabido que tanto la eficiencia de dichos algoritmos heurísticos como los resultados que éstos obtienen mejoran de manera significativa cuando se aplican estrategias de paralelismo. En este punto es importante distinguir entre lo que son estrategias heurísticas y técnicas metaheurísticas. Las primeras son algoritmos muy rápidos que se ajustan de manera fiel a un problema de optimización concreto. Sin embargo, son difíciles de definir para determinados problemas y las soluciones que proporcionan pueden estar lejos del óptimo que se busca. Por otro lado, las metaheurísticas son métodos genéricos cuyo esquema es fácilmente ajustable a la mayoría de problemas, y suelen ofrecer soluciones de alta calidad en tiempos que resultan adecuados en la mayoría de los casos. Por tanto, las técnicas metaheurísticas fueron los algoritmos no-exactos que se eligieron para trabajar en esta tesis.

Sin embargo, otro factor a tener en cuenta es que este tipo de esquemas suele tener un elevado coste computacional que es debido tanto al ajuste de los numerosos parámetros que presentan, como a la ejecución de algunas partes de código donde se hacen exhaustivas búsquedas. Además, cuando se abordan problemas de optimización reales, la complejidad y dimensiones de los mismos hacen que la búsqueda de soluciones óptimas sea una tarea especialmente costosa, por lo que combinar el uso de metaheurísticas con técnicas de paralelismo hace posible que tanto la eficiencia de dichos métodos como los resultados que se obtienen mejoren de manera significativa.

Éste es el contexto en el que se incardina esta tesis doctoral. En concreto, se han estudiado, diseñado y desarrollado un conjunto heterogéneo de metaheurísticas, de manera secuencial y aplicando técnicas de paralelismo, con el fin de resolver un importante problema de optimización dentro del dominio de las Telecomunicaciones. Este problema ha sido el de la asignación automática de frecuencias (FAP, del inglés *Frequency Assigment Problem*), y fue elegido tanto por su complejidad (se trata de un problema NP-Completo [1]) como por su relevancia para la industria del sector, ya que el mundo actual en que vivimos está dominado por las comunicaciones, lo cual provoca que los recursos necesarios para llevar a cabo este proceso sean muy limitados para las empresas que se encargan de ofrecer estos servicios, por lo que se hace necesario que éstas los gestionen de manera eficiente para que sea posible sacar el máximo rendimiento de ellos manteniendo unos niveles altos en la calidad de los servicios que se ofrecen a los usuarios. En este sentido, se considera a la planificación automática de frecuencias como una tarea clave tanto para los operadores GSM (*Global System for Mobile* [2]) actuales como para los operadores futuros, ya que sólo con una planificación de alta calidad que consiga obtener el máximo partido del reducido rango de frecuencias de que disponen los operadores de telefonía actuales, es posible llevar a cabo una comunicación de calidad.

En este punto es importante indicar que la tecnología GSM sigue siendo hoy día la tecnología de comunicaciones móviles más difundida y utilizada. De hecho, a mediados de 2009 los servicios GSM tenían alrededor de 3.5 billones de abonados [3] a través de más de 220 países en el mundo. Estas cifras representan aproximadamente el 80% del mercado mundial de telefonía móvil,

por tanto se espera que GSM aún tenga un papel importante en el mundo de las telecomunicaciones durante bastantes años, ya que además del número de usuarios que aún utiliza esta tecnología, se considera un hecho ampliamente aceptado que las nuevas generaciones de sistemas de comunicación móvil, las tecnologías 3G (*Universal Mobile Telecommunication System* –UMTS [4]) y 4G (IP móvil, WiMAX [5, 6]) coexistirán con las últimas actualizaciones del estándar GSM al menos a medio plazo.

En cualquier caso, debido a su relevancia, el problema de la planificación de frecuencias ha sido muy estudiado en las últimas décadas, por lo que hay publicados numerosos trabajos donde se utilizan diferentes aproximaciones y una gran variedad de modelos matemáticos para su resolución [7, 8, 9]. Sin embargo, la mayoría de estas publicaciones resuelven problemas FAP de tipo *benchmark* [8], los cuales tienen una complejidad menor que los problemas basados en redes reales, donde se consideran requisitos y conceptos que son inherentes a dichas redes [10], como son el elevado número de transmisores que se utilizan como soporte de las comunicaciones o las restricciones en las que se basan las interferencias producidas en la red.

El problema de optimización que surge con el FAP se explica porque las redes de telefonía actuales disponen de un rango de frecuencias muy limitado con el que deben dar servicio al cada vez mayor número de usuarios que utilizan los servicios de dicha red. Este hecho causa que las frecuencias sean utilizadas de manera simultánea en distintos puntos de la misma para que los operadores puedan cubrir todas las comunicaciones que se producen. Sin embargo, este solapamiento de frecuencias provoca interferencias que dificultan, e incluso pueden llegar a anular, dichas comunicaciones. Por esta razón se hace necesario realizar una planificación de frecuencias óptima, con la que se consiga maximizar la cobertura en toda la red manteniendo a la vez unos mínimos aceptables en la calidad de servicio que ésta ofrece. Esta tarea, como ya se ha puesto de manifiesto anteriormente, acarrea una alta complejidad (el FAP es un problema NP-completo), por lo que utilizar algoritmos exactos para resolver instancias reales del problema no es factible. Por el contrario, abordar este tipo de instancias con técnicas basadas en búsquedas heurísticas y metaheurísticas es, si no obligatorio, una de las mejores opciones para conseguir planificaciones de frecuencia de alta calidad en tiempos razonables [11, 12, 13].

Por tanto, la investigación llevada a cabo para esta tesis consiste en estudiar un diverso conjunto de metaheurísticas en combinación con estrategias basadas en paralelismo para su aplicación a la resolución de un problema FAP real. En las siguientes secciones de este capítulo se detallan los objetivos alcanzados y las fases de desarrollo de la presente tesis doctoral. Después se enumeran las contribuciones aportadas por la misma, y finalmente se explica la estructura que se ha seguido en el resto del documento.

1.1. Objetivos y fases

El objetivo principal de esta tesis doctoral es, por una parte, estudiar y desarrollar técnicas metaheurísticas apoyadas en estrategias de paralelismo que sean capaces de obtener soluciones de alta calidad para el problema de la planificación automática de frecuencias aplicado a redes reales de telecomunicaciones; mientras que por otra, analizar los resultados obtenidos con dichas técnicas para diseñar e implementar un sistema de optimización que resuelva dicho problema de manera eficiente y competitiva.

Podemos resumir las fases que se han seguido para la consecución de dicho objetivo en los siguientes puntos:

1. Estudio del problema de optimización con el que se ha trabajado, esto es, el problema de la asignación automática de frecuencias (FAP). En esta fase se realizó un completo estudio bibliográfico con el objetivo de entender adecuadamente los detalles del problema, sus distintas variantes y las posibles técnicas que podían ser aplicadas en su resolución.

2. Modelado de un primer algoritmo heurístico que fuera capaz de abordar y resolver instancias sencillas del problema, de tipo benchmark. Desarrollo de una primera función de coste que determine la calidad de las soluciones generadas con esta primera aproximación.

3. Modificación del algoritmo anterior y de la función de coste asociada para la aplicación del método a instancias reales del problema.

4. Estudio bibliográfico para determinar las estrategias más utilizadas y que aportaran mejores resultados en la resolución de instancias reales del problema FAP.

5. Análisis y selección de una primera metaheurística que resuelva de forma competitiva una instancia real del problema. La estrategia seleccionada debía constituir una apuesta novedosa que a la vez ofreciera buenas características para aplicarse con éxito al problema planteado. Además, tras el desarrollo de esta aproximación se debía incluir un completo ajuste paramétrico con el fin de obtener resultados de alta calidad.

6. Análisis de los resultados y comparación objetiva de los mismos con los obtenidos por otros autores utilizando otras estrategias. Obtención de conclusiones y depuración de la metaheurística desarrollada para maximizar la calidad de los resultados conseguidos.

7. Estudio, selección, desarrollo, ajuste y optimización de un conjunto heterogéneo de metaheurísticas que abordaran la resolución del problema FAP desde varios enfoques. El desarrollo de estos nuevos algoritmos debía apoyarse en la experiencia obtenida durante las fases anteriores de la investigación, por lo que para el diseño de las nuevas estrategias se tuvieron en cuenta las ideas y detalles aprendidos con el desarrollo y depuración de la primera metaheurística estudiada.

8. Análisis de los resultados obtenidos por cada una de las estrategias desarrolladas, estudiando la calidad de las soluciones aportadas en distintos periodos de tiempo y con distintas instancias reales del problema. Comparación de resultados y obtención de conclusiones.

9. Estudio bibliográfico de metaheurísticas paralelas, algoritmos paralelos que trabajaran con metaheurísticas y estrategias paralelas utilizadas para resolver problemas complejos de optimización.

10. Diseño y construcción de un sistema paralelo que sacara partido a la colaboración de diversas metaheurísticas trabajando a la vez. El objetivo de dicho sistema debía ser la mejora de los resultados obtenidos por estas técnicas, tanto en eficiencia como en calidad de las soluciones aportadas.

11. Evaluación, estudio estadístico y comparación con otros autores de los resultados conseguidos por el sistema paralelo desarrollado. Depuración y mejora del mismo.

12. Análisis final de resultados y obtención de conclusiones finales del trabajo realizado. Redacción del presente documento de tesis doctoral.

Como se puede observar, la dinámica básica que se ha aplicado durante las etapas llevadas a cabo en esta tesis ha sido: estudio, desarrollo, análisis y comparación de resultados. De esta manera, cada nuevo paso que se ha dado en la investigación se ha apoyado en la experiencia recabada en las fases anteriores y en un riguroso análisis de los resultados empíricos que se han ido obteniendo en cada fase, validando además dichos resultados mediante análisis estadístico y comparación tanto con nuestros propios avances como con otros trabajos publicados. Por tanto, las aportaciones y conclusiones de esta tesis están respaldadas por la utilización del método científico.

1.2. Contribuciones

Las principales contribuciones de esta tesis doctoral se pueden resumir mediante los siguientes puntos:

- Modelado y ajuste de diversas técnicas metaheurísticas con el objetivo de conseguir resultados de alta calidad en la resolución de instancias reales del problema de optimización abordado en esta tesis.

- Algunas de las metaheurísticas con las que se ha trabajado no habían sido utilizadas previamente en la resolución del problema de la asignación automática de frecuencias, y más aún, en un caso real del problema, por lo que la aplicación de esos algoritmos en la resolución de dicho problema es una contribución original de esta tesis doctoral.

- Diseño e implementación de una novedosa estrategia paralela (hiperheurística paralela basada en metaheurísticas) que tampoco había sido utilizada antes para la resolución de un problema FAP real.

- Hasta donde sabemos, la hiperheurística desarrollada en esta tesis mejora los resultados obtenidos hasta la fecha (tanto en tiempo como en calidad) por cualquier otra aproximación que haya sido utilizada para resolver el problema que ha sido objeto de estudio en esta investigación.

- Tanto el modelo matemático utilizado, como las instancias del problema con las que se han hecho todos los experimentos, han sido obtenidos de la industria de las telecomunicaciones, a través del proyecto de investigación nacional coordinado OPLINK, y más concretamente a través del nodo de la Universidad de Málaga [14], por lo que se puede afirmar que la investigación realizada y las conclusiones alcanzadas en esta tesis pueden ser aplicables a la industria del sector.

- Finalmente, se ha tratado que los resultados y conclusiones que se han ido obteniendo durante el transcurso de esta investigación se hayan ido difundido adecuadamente a la comunidad científica, por lo que la publicación de dichos avances en foros científicos de calidad puede considerarse una contribución importante aportada por esta tesis doctoral. A este respecto, en el Apéndice II de este documento se puede encontrar una lista con las referencias de las publicaciones en las que se sustenta la investigación realizada en esta tesis.

1.3. Estructura de la tesis

Esta tesis doctoral se ha estructurado en siete capítulos, más uno de referencias donde se registran los trabajos a los que se hace alusión en este documento, y dos apéndices donde se incluye, por una parte, un resumen del análisis estadístico realizado a los datos, y por otra, las publicaciones en las que se sustenta la investigación realizada en estos años.

El primero de los capítulos es el presente apartado introductorio, donde se han introducido, por una parte, el problema de optimización con el que se ha trabajado y las técnicas que se han utilizado para su resolución, y por otra, las motivaciones y los objetivos alcanzados en esta tesis. En cuanto al resto de los capítulos del presente documento, su título y descripción son los siguientes:

Capítulo 2. Antecedentes y trabajo relacionado

En esta sección se hace una revisión de las publicaciones que están relacionadas con el trabajo realizado en esta tesis. Algunas de ellas sirvieron de punto de partida para la investigación realizada, mientras que otras se han utilizado como soporte o herramienta de comparación en distintos puntos de la investigación. En cualquier caso, muchos de los trabajos consultados constituyen la base sobre la que comenzó esta tesis. Para ser más específicos, en este capítulo se revisan los trabajos más relevantes referidos al problema de la planificación automática de frecuencias, especialmente los que lo abordan utilizando técnicas metaheurísticas y estrategias basadas en paralelismo. Además, dado que en esta tesis se ha utilizado una hiperheurística paralela, también se ha incluido aquí un estudio bibliográfico acerca de esta técnica y su aplicación en la resolución de problemas de optimización.

Capítulo 3. Fundamentos del problema

En este apartado se explican los fundamentos teóricos del problema de la asignación automática de frecuencias. Para ello, en primer lugar, se realiza una

explicación general del mismo, donde se describen sus características y requisitos utilizando ejemplos sencillos basados en *benchmarks*, y a continuación se exponen los detalles, requisitos específicos y el modelo matemático del caso real que se ha abordado en esta tesis.

Capítulo 4. Metodologías aplicadas en la resolución del problema

Este capítulo describe de manera teórica las estrategias de resolución utilizadas en esta tesis doctoral. Por tanto, en primer lugar se explica de manera introductoria las aproximaciones algorítmicas que podrían aplicarse al problema que se aborda en esta tesis. A continuación se desarrollan los conceptos básicos sobre metaheurísticas así como los de cada una de las técnicas concretas aplicadas en la resolución del problema abordado. Después se introduce la forma de evaluar los resultados obtenidos por las metaheurísticas y finalmente se describen las principales estrategias de paralelismo que se han utilizado en nuestra investigación en combinación con las técnicas metaheurísticas.

Capítulo 5. Diseño y desarrollo de algoritmos

En este capítulo se explica en detalle cómo se ha modelado y resuelto el problema FAP haciendo uso de las técnicas explicadas en el capítulo anterior. Por tanto, esta sección incluye una descripción detallada de la adaptación que se ha llevado a cabo para cada una de las estrategias desarrolladas en esta tesis.

Capítulo 6. Análisis de resultados

En esta sección se describen las condiciones en las que se han realizado las pruebas y los experimentos llevados a cabo en nuestra investigación. A continuación se exponen y analizan los resultados de los ajustes paramétricos y diferentes configuraciones probadas para cada técnica. Por último, se presentan y comparan los datos finales arrojados por cada estrategia. A este respecto, queremos indicar que se ha realizado un análisis comparativo utilizando tanto datos propios, arrojados por los diferentes algoritmos desarrollados, como datos obtenidos con otras técnicas desarrolladas por otros autores y publicadas en la bibliografía.

Capítulo 7. Conclusiones

Finalmente, en esta sección se discuten las conclusiones alcanzadas tras la investigación desarrollada. De este modo, en este capítulo se expondrán, por un lado, las conclusiones alcanzadas tras el análisis de los resultados, y por otro, las principales aportaciones conseguidas con la realización de esta tesis doctoral. Además, en este apartado se trazan posibles líneas de trabajo futuro que se desprenden del trabajo presentado en este documento. Finalmente, se cierra con las conclusiones personales del autor de esta tesis tras los años de investigación realizados.

Capítulo 2

Antecedentes y trabajo relacionado

En este capítulo se realiza un análisis bibliográfico de los trabajos que tienen relación con la investigación desarrollada en esta tesis. Debido al gran número de publicaciones que se encuentran en la bibliografía sobre el problema FAP, en este documento se ha limitado la discusión de las mismas a aquellas que utilizan estrategias similares a las que se emplean en esta tesis, es decir, a los trabajos que resuelven el problema de la asignación automática de frecuencias con técnicas metaheurísticas y a los que utilizan para su resolución estrategias de paralelismo. Además, dado que los mejores resultados de nuestra investigación se han obtenido con una hiperheurística paralela, también se ha incluido aquí un estudio bibliográfico acerca de esta técnica y de su aplicación en la resolución de problemas de optimización.

El FAP [8] es un problema de optimización muy conocido dentro del campo de la Investigación Operativa [7]. Fue introducido por B. H. Metzger a comienzos de la década de los setenta [15], y desde entonces se ha publicado una enorme cantidad de trabajos relacionados con él [16]. Además, es importante señalar que no hay una única versión del problema, ya que dependiendo de los requisitos y objetivos que se persigan en la red de comunicaciones con la que se trabaje, se distinguen varios tipos. Por ejemplo, si se considera a las frecuencias de manera discreta, como unidades, que es la forma en la que eran tratadas a comienzos de la década de los setenta, el objetivo del problema de la asignación de frecuencias es el de minimizar el número de frecuencias que se utilizan en la planificación de una red de comunicaciones específica. A esta variante del FAP se la conoce como MO-FAP, del inglés *Minimum Order Frequency Assignment*

Problem [7], y se considera como una generalización directa del problema de coloración de grafos [7, 8]. Unos años después, a comienzos de los ochenta, el modelo que representaba al FAP fue mejorado, incluyéndose por ejemplo las interferencias que se generan entre frecuencias de canales adyacentes [1]. De esta manera, las frecuencias comenzaron a ser gestionadas como bloques en lugar de como unidades y todo ello condujo a la creación de una nueva variante del FAP a la que se llamó MS-FAP (*Minimum Span Frequency Assignment Problem*) [7], donde el objetivo es minimizar la diferencia entre la frecuencia más baja y más alta que se utiliza en la planificación. Al igual que la versión anterior, se considera que ésta es una generalización del problema de coloreado de grafos, por lo que ambas versiones del problema (la MO-FAP y la MI-FAP) han sido ampliamente estudiadas desde su aparición, y es posible encontrar multitud de trabajos en la bibliografía donde se utilizan diversas técnicas para su resolución. Sin embargo, la mayoría de estas técnicas son heurísticas que, o bien provienen de algoritmos especialmente diseñados para el coloreado de grafos [17-19], o bien se trata de otras estrategias más avanzadas que se utilizan de manera habitual para resolver problemas de optimización, como son las búsquedas tabú o los algoritmos genéticos [20, 21].

Sin embargo, es bien conocido que el tráfico que se genera en las redes de comunicaciones ha experimentado un enorme crecimiento en las últimas dos décadas, por lo que el problema de la planificación de frecuencias ha ido también evolucionando de acuerdo con las exigencias técnicas de los tiempos, y se ha convertido en un problema mucho más complejo del que fue en sus formas primigenias. De esta forma, el objetivo del FAP ha pasado de ser el de minimizar el número de frecuencias que se usa en la planificación de una red (MO-FAP y MS-FAP), a ser el de minimizar las interferencias (MI-FAP –*Minimum Interference-FAP* [7]) o incluso los bloqueos (MB-FAP –*Minimum Blocking-FAP* [7]) que se producen en una red real de telecomunicaciones.

Aunque en los comienzos de la investigación realizada para esta tesis se hicieron pruebas con el conjunto de instancias Philadelphia [8], que trabajan con la versión MS-FAP del problema, pronto nos centramos en la resolución de la variante MI-FAP, ya que es el tipo de problema que se maneja habitualmente cuando se trabaja con instancias reales y actuales del mismo. No daremos más información en este capítulo acerca de la evolución histórica del FAP y de sus

distintas variantes. El lector interesado puede obtener más información de este tema en las referencias [7] y [8].

En cualquier caso, el problema que surge en las redes reales es que éstas disponen de un número muy limitado de frecuencias (normalmente no más de veinte) para soportar las miles de comunicaciones que se generan de manera simultánea entre los abonados a dicha red. Por esta razón se hace necesario que las frecuencias se utilicen múltiples veces para dar cobertura a todas las comunicaciones que se pueden producir. Sin embargo, el uso de la misma frecuencia en distintos puntos de la red causa interferencias cuyo impacto ha de ser minimizado para optimizar la calidad de las comunicaciones que se producen. Este es el objetivo final de la variante MI-FAP, que por simplicidad en la nomenclatura, es el problema al que se hace referencia en esta tesis cuando se habla de manera general del problema de la planificación automática de frecuencias (FAP).

Como se ha indicado anteriormente, el FAP es un problema bien conocido, tanto por su antigüedad como por su importancia, por lo que se pueden encontrar en la literatura trabajos y revisiones muy relevantes [7, 10, 17, 22-25]. En dichos estudios se utilizan numerosas técnicas para la resolución eficiente del problema, si bien cabe destacar que los métodos heurísticos y metaheurísticos son los más comunes [7]. Por el contrario, el uso de algoritmos exactos para su resolución es casi inexistente [26, 27], ya que la aplicación de dichas técnicas es inviable cuando se trabaja con instancias de gran tamaño. La razón es que el problema FAP es muy complejo, de hecho es NP-Completo, por lo que el uso de técnicas no exactas es obligatorio cuando se manejan problemas de tamaño real [7, 11, 12]. De hecho, si se examinan los trabajos que manejan problemas con características similares al que se aborda en esta tesis (instancias reales del problema MI-FAP), se observa que los mejores resultados se obtienen utilizando metaheurísticas [11, 13] que normalmente se combinan con alguna heurística de búsqueda local (algoritmo avaricioso) para mejorar y completar el carácter estocástico de las metaheurísticas [28].

Debido a la enorme cantidad de trabajos que hay publicados donde se aplican metaheurísticas a la resolución del problema que se aborda en esta tesis, nos vemos obligados en este capítulo a limitar el análisis bibliográfico a aquellos que están relacionados directamente con las técnicas que se han utilizado en

nuestra investigación. A este respecto, queremos indicar que tras estudiar una amplia variedad de aproximaciones, finalmente se optó por desarrollar siete metaheurísticas con las que se ha pretendido cubrir un amplio rango de este tipo de estrategias. Como se explicará en el capítulo 4 de esta tesis, las aproximaciones desarrolladas han sido:

- Algoritmo genético (GA –*Genetic Algorithm*) [29, 30]. A tenor de la cantidad de trabajos publicados en la bibliografía, se puede considerar a este clásico algoritmo como el más conocido y utilizado de los algoritmos evolutivos (EA –*Evolutionary Algorithms*) [31]. Su funcionamiento está inspirado en la teoría de la evolución natural, y muchas de las metaheurísticas basadas en población que se han desarrollado después están basadas en él.

- Búsqueda dispersa (SS –*Scatter Search*) [32-35]. Esta metaheurística basada en población es una de las más populares entre la comunidad científica en la actualidad. Además, se ha demostrado como una de las técnicas más eficientes en la resolución del problema FAP, como se demostrará en el capítulo de análisis de resultados de esta tesis.

- Aprendizaje incremental basado en población (PBIL –*Population Based Incremental Learning*) [36, 37]. Esta metaheurística combina estrategias evolutivas con un modelo probabilístico basado en aprendizaje competitivo para la resolución de problemas de optimización.

- Búsqueda de entorno variable (VNS –*Variable Neighbourhood Search*) [38, 39]. Esta estrategia basada en trayectoria cambia de entorno (o de vecindario) durante la búsqueda del óptimo de forma que se reduzcan las posibilidades de caer en mínimos locales.

- Búsqueda local iterativa (ILS –*Iterated Local Search*) [40, 41]. Metaheurística muy simple que basa su funcionamiento en practicar en cada iteración una perturbación y una operación de búsqueda local sobre la solución del problema con la que se está operando.

- Procedimiento de búsqueda avariciosa, aleatoria y adaptativa (GRASP –*Greedy Randomize Adaptive Search Procedure*) [42, 43]. Este método

combina procedimientos constructivos, con los que se genera una solución factible para el problema, con una estrategia de búsqueda local, con el que se mejora la solución generada en la etapa de construcción.

- Algoritmo basado en colonia de abejas (ABC –*Artificial Bee Colony*) [44-46]. Metaheurística que basa su funcionamiento en la inteligencia colectiva, y más concretamente, en la inteligencia de las abejas recolectoras de miel. En el algoritmo ABC se distinguen varios tipos de operaciones de búsqueda de acuerdo con los comportamientos que tienen los distintos tipos de abeja dentro de una colonia.

Por otra parte, en esta tesis se ha desarrollado una hiperheurística paralela como técnica de mejora para obtener soluciones de alta calidad que superen los resultados obtenidos por las metaheurísticas cuando se manejan instancias de gran tamaño. Por tanto, este capítulo también incluye un estudio bibliográfico de los trabajos en los que esta técnica ha sido aplicada a la resolución de problemas reales de optimización. En este sentido, es cierto que el número de trabajos que se pueden encontrar en la bibliografía acerca del uso de estrategias paralelas con las que se mejora el rendimiento de metaheurísticas es enorme, tal y como demuestran numerosos trabajos de referencia e importantes revisiones sobre la temática [47-54]. Además, muchos de estos trabajos revelan estrategias paralelas que son combinadas con técnicas metaheurísticas para resolver el problema de la planificación automática de frecuencias [7, 10, 25, 55-65]. Sin embargo, una hiperheurística basada en un conjunto heterogéneo de metaheurísticas representa una aplicación innovadora en la resolución del FAP, ya que hasta donde sabemos, no ha sido utilizada con anterioridad en la resolución del problema abordado en esta tesis.

En conclusión, en las siguientes dos secciones se analizarán, por un lado, los trabajos donde se utilizan técnicas hiperheurísticas para la resolución de problemas reales de optimización; y por otro, las publicaciones más destacadas donde se aplican las técnicas metaheurísticas que se han incluido en esta tesis para resolver el problema de la planificación automática de frecuencias.

2.1. Trabajos relacionados con hiperheurísticas

Antes de comenzar la revisión de los trabajos que incluyen hiperheurísticas como técnicas de resolución de problemas, es importante señalar que la propia idea de *hiperheurística* es bastante reciente. De hecho, fue propuesta a finales de los años noventa por *Hart*, *Ross* y *Nelson* [66]. Las hiperheurísticas se pueden definir como "heurísticas para seleccionar heurísticas", y básicamente se diferencian de las metaheurísticas en el espacio de búsqueda en el cual operan. Así, mientras que las metaheurísticas realizan la búsqueda del óptimo entre las soluciones que resuelven un determinado problema, las hiperheurísticas tienen por espacio de búsqueda un conjunto de heurísticas de entre las cuales se debe elegir la más apropiada en cada momento [67-70]. Por tanto, el trabajo de una hiperheurística consiste en encontrar la estrategia o configuración más adecuada ante una situación determinada, y no en resolver directamente el problema (Figura 2.1). Esta idea ha sido ampliamente aplicada a la resolución de una gran variedad de problemas de optimización en diversos campos de investigación, principalmente en áreas de planificación y gestión de recursos.

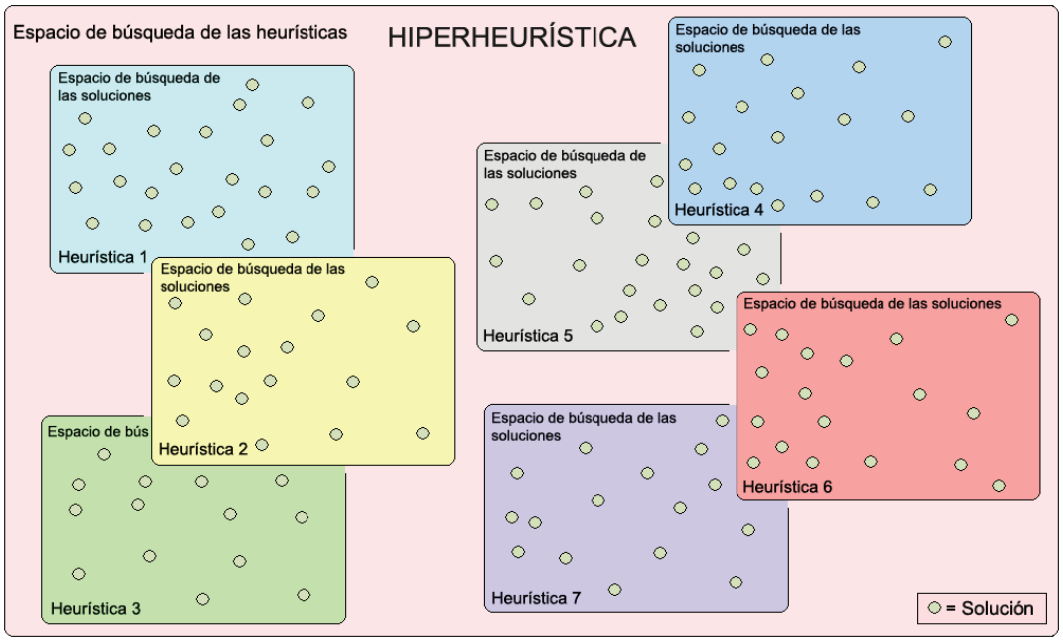

Figura 2.1. Espacios de búsqueda donde trabajan las heurísticas y la hiperheurística

De esta forma, existe una gran variedad de problemas reales que son resueltos utilizando hiperheurísticas. Por ejemplo, en la bibliografía destacan trabajos recientes sobre: gestión de horarios docentes [71-80], empaquetado [81-85], planificación de envíos y gestión de producción [86, 87], distribución de espacio e inventario [88-90], asignación de horarios en hospitales [77, 91, 92], planificación de personal [93-95], encaminamiento en fibra óptica [96], problemas industriales [97, 98], configuración de protocolos de comunicaciones [99], posicionamiento de antenas [100], o incluso configuración dinámica de estrategias evolutivas [101-103]. Además, se han encontrado dos trabajos en los que se ha abordado el problema de la asignación de frecuencias mediante hiperheurísticas [62, 63], sin embargo, estos trabajos se diferencian totalmente del realizado en esta tesis, ya que resuelven instancias de tipo *benchmark* de la variante MS-FAP del problema, mientras que nuestra investigación se ha centrado en instancias reales de la variante MI-FAP, lo cual representa una tarea mucho más compleja.

Por otro lado, las hiperheurísticas son estrategias que pueden ser desarrolladas utilizando una gran variedad de técnicas. En concreto, se han publicado trabajos donde el diseño de la hiperheurística está basado en: razonamiento basado en casos de uso (CBR –*Case Base Reasoning*) [74, 78], estrategias de búsqueda local o búsqueda tabú [77, 91, 101], aproximaciones basadas en teoría de grafos [72, 73], algoritmos genéticos [82, 92, 94, 101], búsqueda dispersa [79], colonia de hormigas [81, 96], enfriamiento simulado [89, 98], algoritmos meméticos [85], técnicas basadas en búsquedas de entorno variable [95], aproximaciones multi-objetivo [90] o estrategias probabilísticas tipo Montecarlo [97].

En conclusión, una hiperheurística es una aproximación heurística que puede ser desarrollada utilizando un amplio rango de técnicas y es aplicable a multitud de problemas de optimización. Sin embargo, a pesar de la gran cantidad de trabajos consultados, no se ha encontrado ninguna publicación donde esta estrategia haya sido utilizada para resolver el problema abordado en esta tesis (MI-FAP), o donde el diseño de la hiperheurística sea similar al expuesto en esta tesis. En todos los estudios analizados las hiperheurísticas se apoyan en búsquedas heurísticas sencillas, o en distintas variaciones de una única metaheurística, mientras que la hiperheurística desarrollada en esta tesis

está basada en siete metaheurísticas diferentes, por tanto, el trabajo desarrollado en esta tesis representa una contribución tanto en el dominio del problema como en el de la solución aplicada para su resolución.

2.2. Metaheurísticas y el FAP

En esta sección se analizan los trabajos relacionados con las metaheurísticas con las que se ha trabajado en esta tesis (PBIL, SS, GA, VNS, ILS, GRASP y ABC). Además, dado que todas ellas han sido utilizadas en un sistema paralelo con el fin de mejorar las soluciones aportadas en la resolución del problema FAP, en esta sección también se ha incluido un estudio bibliográfico de las estrategias paralelas que han sido aplicadas en la resolución del problema de la asignación automática de frecuencias.

El FAP y los algoritmos evolutivos

Los algoritmos evolutivos [31] han sido ampliamente utilizados de manera histórica en la resolución del problema FAP. Prueba de este hecho es la enorme cantidad de trabajos existentes en la bibliografía, algunos de de los cuales son ya considerados como clásicos [59, 60, 105-108] mientras que otros son bastante recientes [109-111]. Además, se pueden encontrar varios tipos y adaptaciones de este tipo de algoritmos para la resolución de varias versiones del problema FAP, por lo que el número de trabajos en este campo es realmente extenso. Por esta razón, en este apartado se reducirá el análisis a aquellos trabajos que tienen una relación directa con la solución aportada en esta tesis. Se discutirán, por tanto, las publicaciones donde se aborda la resolución de la variante MI-FAP del problema utilizando una versión híbrida del algoritmo evolutivo. En nuestro caso, dicha versión híbrida combina un algoritmo genético con una heurística de búsqueda local con la que se mejora el comportamiento estocástico de la metaheurística. La Tabla 2.1 resume las principales

características de los trabajos más relevantes que abordan el MI-FAP mediante algoritmos evolutivos híbridos. Para cada contribución de la tabla se especifica el año de la misma, la referencia bibliográfica, el método con el que se combina el EA, y las instancias utilizadas (incluyendo su tamaño) para realizar los experimentos (cuando esta información venía reflejada en la publicación).

Tabla 2.1. Algoritmos evolutivos híbridos utilizados para resolver el problema de la asignación automática de frecuencias (versión MI-FAP)

Año de publicación	Referencia	Método de mejora	Instancia (tamaño)
2001	[112, 113]	Avaricioso, TS	Propietaria (5700 TRXs)
2002	[64]	TS	Propietaria (639 TRXs)
2003	[114]	Avaricioso	Propietaria
2004	[110, 115]	MDP	Propietaria (5700 TRXs)
2005	[116]	Red neuronal	Philadelphia
2006	[117]	Avaricioso	Generada (1667 TRXs)
2007	[118, 119]	Avaricioso	Propietaria (2612 TRXs)
2008	[120]	Avaricioso	Propietaria (2612 TRXs)
2010	[121, 122]	Avaricioso	Propietaria (2612 TRXs)

A la vista del número de trabajos publicados en los últimos diez años, se puede concluir que este tipo de estrategias son muy adecuadas para resolver el problema que se aborda en esta tesis. Además, en la Tabla 2.1 se puede observar que este flujo de publicaciones ha sido constante, por lo que como primera conclusión se puede extraer el interés de los investigadores en esta estrategia, gracias a lo cual los EAs híbridos han ido mejorando y creciendo año tras año. Por otro lado, es también notorio que el uso de algoritmos avariciosos, como las búsquedas locales (LS –*Local Search*) son las estrategias que más se utilizan en combinación con los algoritmos evolutivos, ya que complementan el comportamiento aleatorio de éstos añadiendo información específica del dominio del problema. En este sentido, se puede afirmar que este tipo de técnicas, que son denominadas, según la gramática de Talbi [28, 123], esquemas LTH (*Low-level Teamwork Hybrid*), son especialmente efectivas para resolver el problema FAP. Sin embargo, la estrategia de más bajo nivel en esta combinación no es siempre una búsqueda local, (en este caso se denomina EA(LS)). Hay una gran variedad de métodos de mejora que han sido utilizados para enriquecer el

comportamiento de los algoritmos evolutivos, como son los procesos de decisión de Markov (MDP –*Markov Decision Processes*) [110, 115], las búsquedas tabú (TS –Tabu Search) [64, 112], o las redes neuronales [116]. Respecto a las instancias del problema utilizadas, las hay de dos categorías: la primera, instancias tipo *benchmark* o generadas automáticamente, como son las instancias Philadelphia, CALMA o CELAR [8], y por otro, las instancias propietarias donde los datos no son normalmente de dominio público, lo cual dificulta la tarea de comparación entre diferentes autores.

El FAP y SS, PBIL y ABC

Comparado con los trabajos que se pueden encontrar en los que el FAP es abordado mediante algoritmos evolutivos, el número de publicaciones en los que el problema es resuelto mediante otras metaheurísticas es bastante limitado. Si comenzamos el análisis con la búsqueda dispersa (SS) [32-35], debemos decir que esta estrategia fue usada en primer lugar para resolver el problema de coloreado de grafos [124], pero también ha sido aplicada al problema FAP en [120, 125-127]. En todos esos trabajos ha colaborado el autor de esta tesis, por lo que se maneja el problema y las instancias reales que se describirán en próximos capítulos de este documento. Además, SS se encuentra en todos los casos combinado con un método de búsqueda local para mejorar los resultados, que debemos decir que son de muy alta calidad, tal y como se discutirá en los sucesivos capítulos de esta tesis.

Con respecto a PBIL [36, 37], el uso de esta metaheurística para resolver el FAP aparece descrito en los trabajos [57, 58, 126, 128-131]. De éstos, sólo en [129] se utiliza la metaheurística para resolver una instancia tipo *benchmark* del problema, la de Philadelphia. Este trabajo representa los inicios de nuestra investigación, ya que en el resto de trabajos publicados se utiliza una versión real del problema. En concreto, las referencias [128, 131] explican los primeros pasos en la investigación con la instancia real, por lo que los resultados que se obtienen son bastante pobres. En [130] se realiza un estudio de las diferentes variantes de la metaheurística, concluyéndose que una versión híbrida de la

versión estándar y un algoritmo de búsqueda local era la mejor aproximación para resolver el problema (idea que luego se aplicó a otras metaheurísticas). El trabajo realizado en [126] consistió en un estudio comparativo de los resultados arrojados con PBIL y los primeros que se obtuvieron con SS, tras el cual se llegó a la conclusión de que la búsqueda dispersa se adaptaba mejor al FAP que PBIL. Finalmente, en [57, 58] se explica una versión paralela de PBIL utilizando computación clúster para mejorar los resultados publicados en [128, 131].

En cuanto a la estrategia basada en colonias de abejas (ABC) [44, 46], actualmente sólo se encuentra publicado el trabajo [121], donde se analiza el comportamiento de esta técnica junto con la búsqueda dispersa y un algoritmo evolutivo. Sin embargo, existen otros trabajos donde el problema FAP ha sido abordado utilizando otras metaheurísticas inspiradas en el comportamiento de los insectos. En concreto, destacan los trabajos [119, 133], donde se aplica la optimización basada en colonias de hormigas (ACO –*Ant Colony Optimization* [132]) en la resolución del problema que se estudia en esta tesis.

El FAP y el GRASP, ILS y VNS

El uso de metaheurísticas basadas en trayectoria para la resolución del problema que se aborda en esta tesis es bastante anecdótico. El algoritmo GRASP [42, 43] es el que presenta mayor número de ocurrencias en la bibliografía. Destacan los trabajos referenciados en [56, 134-136]. Todos ellos combinan la utilización de la metaheurística con otra heurística de búsqueda intensiva para mejorar los resultados que se obtienen. En concreto, el trabajo publicado en [56] explica el desarrollo del algoritmo utilizando una aproximación maestro-esclavo basada en computación grid, cuyos resultados serán discutidos en el capítulo de análisis de resultados de este documento. En cuanto a [134-136], presentan estudios con instancias tipo *benchmark*, ya que en [134, 135] los datos de las instancias están automáticamente generados, mientras que el caso del trabajo [136] utiliza las instancias Philadelphia [8] para realizar todos los experimentos.

Por otro lado, en referencia a los algoritmos derivados de heurísticas basadas en búsquedas locales, es importante señalar que su uso en combinación

con otras metaheurísticas más complejas es bastante común, ya que complementa el comportamiento estocástico que éstas presentan y permite realizar búsquedas más potentes. Además, existen trabajos en la bibliografía donde se utiliza alguna variante de búsqueda local para resolver el FAP, como es la LSHR (*Local Search with Heuristic Restart*) [120, 122], sin embargo, no se han encontrado apenas trabajos en los que se apliquen los algoritmos VNS [38, 39] e ILS [40, 41] en la resolución del FAP. Hasta donde sabemos, sólo existe algún trabajo con VNS, como el descrito en [137], donde se resuelve el FAP aplicando versiones multi-objetivo del algoritmo. En cambio, sí que se han encontrado publicaciones donde se aplican las citadas metaheurísticas para resolver el problema de coloreado del grafos [138, 139], lo cual aportó un motivo para que nos fijáramos en dichos algoritmos, ya que como se ha mencionado anteriormente ambos problemas están relacionados.

Estrategias de paralelismo utilizadas con el FAP

Para terminar el estudio de trabajos relacionados con la investigación realizada en esta tesis, se quiere incluir en esta sección una breve reseña de aquellos estudios donde se han utilizado técnicas de paralelismo en combinación con otras estrategias para resolver el problema de la asignación automática de frecuencias. El número de contribuciones existentes a este respecto es bastante elevado. De hecho, en la bibliografía se pueden encontrar trabajos muy completos donde se explica con amplio detalle la aplicación de varias aproximaciones paralelas con el fin de mejorar las soluciones aportadas de manera secuencial cuando se trabaja con el problema FAP [10, 55]. Por tanto, esta sección se ha limitado al análisis de aquellos trabajos que resuelven la variante del FAP con la que se trabaja en esta tesis (MI-FAP). Es importante señalar en este punto que todos los trabajos encontrados combinan la aplicación de una estrategia paralela con una o varias metaheurísticas. De esta manera, en [57, 58] el problema FAP se resuelve mediante computación clúster (con MPI – *Message Passing Interface* [140]) y PBIL. En esos trabajos se aplicaron varias técnicas paralelas, como son el modelo de islas, el paradigma maestro-esclavo o el paralelismo de grano fino (paralelismo a nivel de población) para mejorar los

resultados obtenidos con la versión secuencial del algoritmo. En cualquier caso, el modelo paralelo que se propone en esta tesis, la hiperheurística paralela, mejora los resultados obtenidos en esos estudios, como se mostrará en el correspondiente capítulo de análisis de resultados de esta tesis.

Por otro lado, la computación grid también ha sido empleada para resolver el problema FAP. Así, en [56] se obtienen soluciones de alta calidad para el problema utilizando una versión paralela de GRASP basada en un paradigma maestro-esclavo sobre una grid, mientras que el trabajo expuesto en [141] consigue excelentes resultados utilizando un algoritmo genético basado en grid. Todos estos trabajos serán discutidos y comparados en la sección de análisis de resultados de esta documentación. Finalmente, el trabajo expuesto en [142] ilustra el uso de computación grid, en combinación con un algoritmo evolutivo, para manejar instancias FAP de grandes dimensiones, aunque en este caso concreto el estudio se centra más en la aplicación misma del paralelismo que en la propia resolución del problema.

2.3. Resumen

Después de realizar un completo estudio donde se han revisado los trabajos que están relacionados con las técnicas de optimización que se aplican en esta tesis en la resolución del problema FAP, podemos afirmar que, hasta donde sabemos, la mayoría de ellas, salvo los EAs y SS, no habían sido aplicadas previamente por otros autores al problema abordado en esta investigación. De hecho, el autor de esta tesis ha colaborado de manera activa en muchas de las contribuciones que hay publicadas sobre este campo en la bibliografía, contribuyendo a la difusión de la utilización de este tipo de estrategias para resolver el problema de optimización propuesto en esta tesis.

Por otra parte, también se puede concluir que hasta donde conocemos, la técnica paralela utilizada en la resolución del problema proporciona, por un lado, los mejores resultados publicados hasta la fecha en la resolución del problema que se maneja en esta tesis, y por otro, representa un trabajo innovador, ya que

no se ha encontrado ningún trabajo donde se utilice esta técnica para la resolución del problema que se aborda en esta tesis.

Capítulo 3

Fundamentos del problema

En esta sección se explican los fundamentos teóricos del problema abordado en esta tesis. El capítulo comenzará con una explicación genérica del problema, definiendo las variantes más importantes del mismo y diferenciando entre los distintos tipos de instancias que se pueden utilizar para trabajar con él. A continuación se abordará de manera introductoria el problema de la planificación de frecuencias aplicado a redes GSM, que es el tipo de problema en el que se ha centrado esta tesis. Finalmente se expondrá la formulación matemática que se ha utilizado para modelar el problema real.

3.1. El problema de la asignación de frecuencias. Definición, variantes y benchmarks

El problema de la asignación automática de frecuencias (FAP –Frequency Assignment Problem), también conocido como planificación automática de frecuencias (AFP –*Automatic Frequency Planning*) o problema de la asignación de canales (CAP –*Channel Assignment Problem*) es uno de los problemas de optimización combinatoria más importantes dentro del dominio de las telecomunicaciones. Consiste en planificar las frecuencias disponibles en una red de telefonía para ofrecer el máximo número de comunicaciones entre los usuarios de la red, a la vez que se minimizan las interferencias que se producen.

La planificación de frecuencias es el último paso en el diseño de una red GSM. Antes de afrontar este problema, el diseñador de la red tiene que tratar

con cuestiones previas, como la localización y configuración de las antenas que darán cobertura a la red (a este problema se le denomina problema de la planificación de celdas –ACP, *Automatic Cell Planning* [143, 144]). También se debe decidir el número y orientación de los sectores que tendrá cada antena y la distribución de TRXs (transmisores-receptores) dentro de éstos [143], que son los elementos encargados de realizar la comunicación. El número de TRXs que se instalarán en cada sector dependerá de la demanda de tráfico que éste deba soportar. Tras estas operaciones se debe realizar una planificación eficiente de las frecuencias de que dispone la red. La planificación de frecuencias consiste en asignar un canal (o frecuencia) a cada TRX [10], aunque no se puede hablar de que exista un único problema de la asignación de frecuencias. Dependiendo del tipo y requerimientos de la red con la que se trabaje, el problema tendrá una definición y unos parámetros distintos.

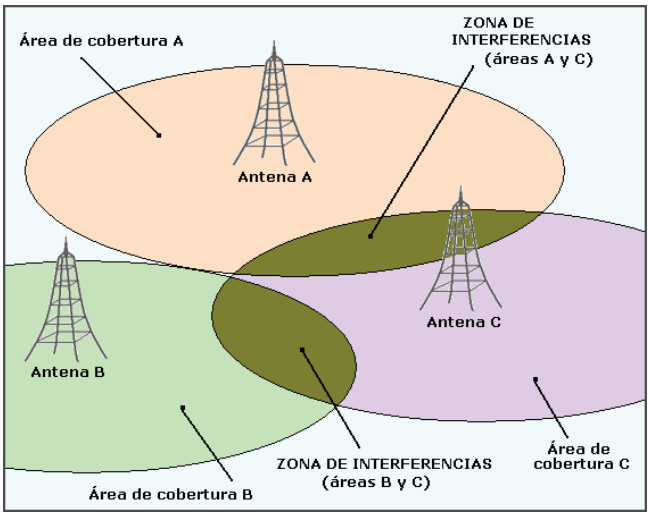

Figura 3.1. Cobertura ofrecida por una red de tres antenas, donde se pueden producir interferencias si las antenas A y C o B y C utilizan frecuencias solapadas

En cualquier caso, el origen del FAP viene marcado por el enorme desarrollo que han tenido los servicios inalámbricos en las últimas décadas. Este rápido desarrollo ha provocado que el espectro de frecuencias disponibles para las operadoras de telefonía, que es muy limitado, sea claramente insuficiente para cubrir toda la demanda necesaria. Por esta razón se ha hecho imprescindible la reutilización del espectro de frecuencias en una misma red de comunicaciones. Sin embargo, esta reutilización de frecuencias conlleva

importantes desventajas, como la pérdida de calidad en determinadas comunicaciones debido a las interferencias producidas por el uso solapado del mismo rango de frecuencias. La Figura 3.1 explica este fenómeno de manera gráfica.

El objetivo de la planificación de frecuencias es el de encontrar un equilibrio entre el número de frecuencias reutilizadas y la pérdida de calidad en la red, de manera que se intenta maximizar las frecuencias que se utilizan varias veces en las comunicaciones minimizando (o anulando si es posible) las interferencias que se generan por esta reutilización. No existe una única versión del problema de la asignación automática de frecuencias. De hecho, tanto el modelo matemático utilizado para representar el problema [7, 10] como la técnica de resolución aplicada para resolverlo variarán de forma significativa dependiendo del escenario abordado y del objetivo concreto que se persiga. Algunos ejemplos de aplicaciones basadas en comunicaciones inalámbricas donde se debe resolver el problema FAP son: las comunicaciones de radio y televisión, las redes de telefonía móvil, los sistemas de comunicaciones móviles por satélite o algunos sistemas de radio (más información en [7, 8]).

Las aplicaciones anteriormente citadas tienen una característica en común: a la hora de establecer una conexión inalámbrica para realizar una comunicación, tanto el emisor como el receptor de la misma deben estar configurados con la misma frecuencia. Si este hecho se generaliza para todas las conexiones que se producen en una red compleja, el problema resultante es el de la planificación automática de frecuencias.

La banda de frecuencias [f_{min}, f_{max}] de que dispone cualquier proveedor de comunicaciones inalámbricas es normalmente fragmentada en un conjunto de canales todos con el mismo ancho de banda Δ. Por esta razón, los canales disponibles son normalmente numerados de 1 a N, donde $N = (f_{max}-f_{min})/\Delta$, y el rango de canales disponibles se dice que está dentro del dominio $D = \{1…N\}$. Así, por cada canal del dominio, D, se puede comunicar información entre un emisor y un receptor, por lo que se establece una relación directa entre el número de canales de una red y el rango de frecuencias que se puede asignar.

En cuanto a las interferencias, éstas se definen mediante el ratio señal-interferencia (o relación señal-ruido) que se recibe en el receptor de la

comunicación (que debe poder reconocer la señal que le llega). El ruido proviene de otras señales transmitidas en una frecuencia cercana a la que difunde la señal, de manera que ambas frecuencias introducen ruido en la señal de la otra. Si se diera el caso de que dos señales distintas se transmitieran por la misma frecuencia, éstas se solaparían y harían ambas señales irreconocibles. En caso de utilización de frecuencias cercanas, sólo se producirán interferencias (de mayor o menor intensidad dependiendo de la cercanía de las frecuencias y de la configuración y requerimientos de la red). Este concepto está representado gráficamente en la Figura 3.2.

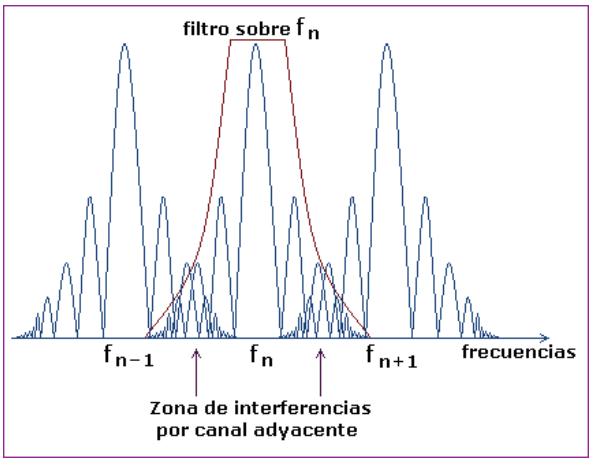

Figura 3.2. Ejemplo de interferencias por canal adyacente

En resumen, el problema FAP (*frequency assignment problem* o problema de asignación de frecuencias) consiste en que dadas una serie de antenas de comunicación (las cuales poseen un rango de frecuencias disponibles en las que transmitir), se desea transmitir por ellas un número concreto de comunicaciones simultáneas sin que se produzcan conflictos entre ellas. Dichos conflictos pueden surgir cuando en una antena se elijen las mismas frecuencias para dos comunicaciones distintas (produce un solapamiento en las comunicaciones), o si no se deja un mínimo de distancia entre las frecuencias elegidas (produce interferencias dentro de la misma antena), o incluso cuando antenas cercanas usan frecuencias que interfieren unas con otras debido a características intrínsecas de la red de comunicaciones (produciéndose en este caso interferencias entre antenas cercanas).

Por tanto, podemos decir que, de manera general, el problema de asignación de frecuencias consiste en:

1. Asignarle a cada antena el número de frecuencias necesario para que pueda transmitir todas las comunicaciones que debe realizar.
2. Hacer que las frecuencias asignadas a cada antena no interfieran entre sí dentro de la misma antena, es decir, que no existan frecuencias repetidas que causen solapamiento (esto es inaceptable en cualquier planificación), y que se asigne un rango de frecuencias que respete la distancia entre las mismas para que no se produzcan interferencias internas (o para minimizarlas si no es posible anularlas).
3. Hacer que las frecuencias elegidas para cada antena de la red no interfieran (o interfieran lo mínimo) con las elegidas para las antenas vecinas (y así evitar las interferencias entre distintas antenas de la red).

Además, tal y como se indicó en el capítulo de *Antecedentes y trabajo relacionado*, existen varias versiones del FAP dependiendo del objetivo que se persiga en el problema. Así, dependiendo de las circunstancias y necesidades de la red con la que se trabaje, el objetivo de la asignación de frecuencias será el de aprovechar el rango de frecuencias lo mejor posible para salvaguardar el máximo número de ellas libres para futuras comunicaciones (MO-FAP y MS-FAP) o bien el de realizar la planificación lo mejor posible para minimizar las interferencias (MI-FAP) o incluso los bloqueos (MB-FAP) que se producen en una red con escasez de recursos, lo cual suele ser la situación habitual. El lector puede encontrar más información acerca de las variantes del problema FAP en el capítulo 2 de esta documentación y en la referencia [7].

Para concluir con la explicación teórica del problema, veamos un sencillo ejemplo ilustrativo de problema FAP de tipo MO, donde se busca utilizar un número de frecuencias mínimo para dar soporte a una determinada red. Supongamos que se dispone de 4 antenas. Cada antena dispone de 11 canales de frecuencia disponibles para emitir y recibir información. Los requerimientos del problema establecen que para las antenas 1, 2 y 3 se deberá transmitir solamente una comunicación, mientras que para la antena 4 se desean transmitir 3 comunicaciones. Es decir, las especificaciones del problema son las siguientes:

- Nº de antenas: 4
- Canales disponibles: 11 por antena: $F = \{1, 2, \ldots 11\}$
- Vector de requisitos: 1 comunicación para las antenas 1, 2 y 3
  3 comunicaciones para la antena 4.

Además, las restricciones de separación entre canales vienen dadas por la siguiente matriz de separación de canales o matriz *CSM* (*Channel Separation Matrix*):

$$CSM = \begin{bmatrix} 5 & 4 & 0 & 0 \\ 4 & 5 & 0 & 1 \\ 0 & 0 & 5 & 2 \\ 0 & 1 & 2 & 5 \end{bmatrix}$$

Cada elemento CSM_{ij} define las restricciones de separación que deben tener los canales de la antena *i* con los canales de la antena *j* si se quieren evitar las interferencias. De esta manera, para que no se produzcan interferencias dentro de la antena 1, se busca la primera fila y la primera columna de la matriz (CSM_{11}) y se comprueba que en esa antena, la restricción de separación para evitar interferencias es de 5, por lo que las frecuencias deben separarse 5 unidades entre sí para evitar interferencias en la antena 1. Para comprobar las restricciones de la antena 1 con la 2, se va a la celda $CSM_{12} = 4$, por lo que han de distar 4 canales para evitar interferencias.

En base a todos los datos anteriores se puede ofrecer una solución (que no es única) al problema según se puede observar en la Figura 3.3.

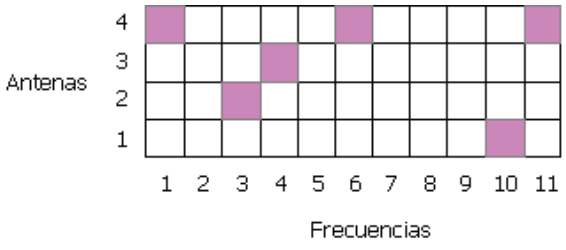

Figura 3.3. Posible solución para el problema descrito. Las celdas sombreadas indican la asignación de la frecuencia señalada por la columna para la antena que se especifica en la fila

Según la Figura 3.3, a la antena número 1 se le ha asignado la frecuencia 10, a la 2 la frecuencia 3, a la 3 la frecuencia 4 y a la antena 4 las frecuencias 1,

6 y 11. Como se puede comprobar examinando los requerimientos del problema, esta es una solución válida (aunque no es la única), ya que se ha satisfecho el vector de requisitos $R = \{1, 1, 1, 3\}$, y se han respetado todas las restricciones indicadas en la matriz *CSM*. Sin embargo, en los problemas reales, el problema a resolver es otro, ya que no hay suficientes frecuencias para cubrir todas las comunicaciones sin provocar interferencias, por lo que el objetivo es minimizar dichas interferencias (MI-FAP), en lugar de minimizar el número de frecuencias que se utilizan.

Por otro lado, para estudiar el problema de la asignación de frecuencias, existen una serie de instancias donde se representan topologías de redes de comunicaciones y las especificaciones necesarias para conocer las restricciones que se presentan en la red para realizar la planificación de frecuencias. Básicamente existen dos tipos de instancias. Por un lado están las instancias que se encuentran en la literatura como instancias de prueba o de tipo *benchmark* [145], y por otro están las instancias que representan problemas reales y que son utilizadas por las empresas de telecomunicaciones para realizar la planificación de frecuencias real de una determinada zona.

En las primeras etapas de la investigación se realizaron pruebas con diversas instancias tipo benchmark, ya que éstas son fácilmente accesibles y existen muchos trabajos publicados en las que éstas son utilizadas, sin embargo, es bien sabido que tanto la complejidad como la repercusión y relevancia de las instancias reales es mucho mayor, por lo que esta tesis se centró finalmente en la resolución de instancias reales del problema. Entre las instancias benchmark del FAP cabe destacar el conjunto de instancias Philadelphia, ya que con estas instancias se consiguió la publicación [129] en las primeras fases de nuestra investigación, por lo que se ha considerado relevante introducirlas en este capítulo de la tesis.

Por último, señalar que las soluciones que se pueden encontrar para los problemas FAP no son únicas, ya que se pueden conseguir varias soluciones válidas de alta calidad respetando las restricciones concretas de cada problema. Asimismo, es importante apuntar que el FAP es un problema de gran complejidad (es NP-Completo), y más aún, si se trabaja con datos reales la obtención de soluciones óptimas supone una tarea de gran complejidad. Es por esto que el uso de estrategias inexactas basadas en inteligencia artificial, como

las metaheurísticas en las que se centra esta investigación, constituyen la mejor opción de resolución del problema, como se puso de manifiesto en el capítulo de *Antecedentes y trabajo relacionado* de este documento (ver Capítulo 2).

El conjunto de instancias Philadelphia

El conjunto de instancias *Philadelphia* [145] es uno de los bancos de prueba del problema FAP más representativos de la literatura. Adopta el nombre de Philadelphia porque se creó para resolver la planificación de frecuencias de la cuidad norteamericana que tiene el mismo nombre. El conjunto de problemas de esta instancia son del tipo MS-FAP, que como ya se ha comentado, tratan de minimizar el espacio de frecuencias usadas en la planificación, a fin de dejar el mayor rango posible de frecuencias disponibles para futuras asignaciones. Las restricciones que debe cumplir el modelo que resuelva el problema son las siguientes:

- Restricciones "co-antena" (*co-site constraints*): Las frecuencias asignadas a la misma antena deben respetar una distancia mínima.
- Restricciones "co-canal" (*co-channel constraints*): Frecuencias iguales sólo pueden ser asignadas a antenas diferentes que respeten una mínima distancia la una de la otra.
- Restricciones de "canal adyacente" (*adyacent-channel constraints*): Frecuencias adyacentes no pueden ser asignadas a antenas adyacentes de manera simultánea. Es necesario que se respete una distancia especificada entre las frecuencias que se asignan a antenas adyacentes.

Si alguna de estas restricciones no se cumple, se producirán interferencias, haciendo imposible la utilización de ciertas frecuencias. Las reglas de compatibilidad electromagnética (EMC) para representar una red de n celdas son descritas por una matriz simétrica de n filas y n columnas denominada *matriz de interferencias (C).* Cada elemento no diagonal de la matriz $C_{ij} \in C$ representa la distancia de separación mínima en el dominio de la frecuencia entre la frecuencia

asignada a la celda *i* y la frecuencia asignada a la celda *j*. $C_{ij}=0$ indica que las celdas de la red *i* y *j* pueden utilizar la misma frecuencia. Las celdas C_{ii} representan la separación de distancia mínima que deben tener dos frecuencias asignados a la celda *i*. Esto representa la restricción "co-antena" de la que se ha hablado antes. Pero además, para resolver el problema FAP es necesario conocer las demandas de cada antena. Esto se representa mediante un vector de *n* posiciones que es llamado *vector de requisitos (demand vector, D)*. Cada elemento $d_i \in D$ representa el número de frecuencias que deben ser asignadas a la antena *i*. El problema FAP que se aborda en las instancias Philadelphia trata de encontrar una asignación de frecuencias óptima libre de conflictos. Sin embargo, es necesario otro parámetro más además de la matriz de interferencias y el vector de requisitos para la variante MS-FAP con la que se trabaja con este conjunto de instancias, y es el parámetro denominado *lower bound (lb)* [146] y representa el valor más pequeño de la máxima frecuencia para todo *i* y para todo *k* en f_{ik} para que no se genere ninguna interferencia. El parámetro *lb* indica que las frecuencias pueden tomar un valor entre *1* y *lb* sin que por ello se incumpla ninguna restricción. Por tanto, a partir del parámetro *lb* se puede calcular el mínimo rango de frecuencias necesarias para resolver el problema (también llamado *span* = lb – 1).

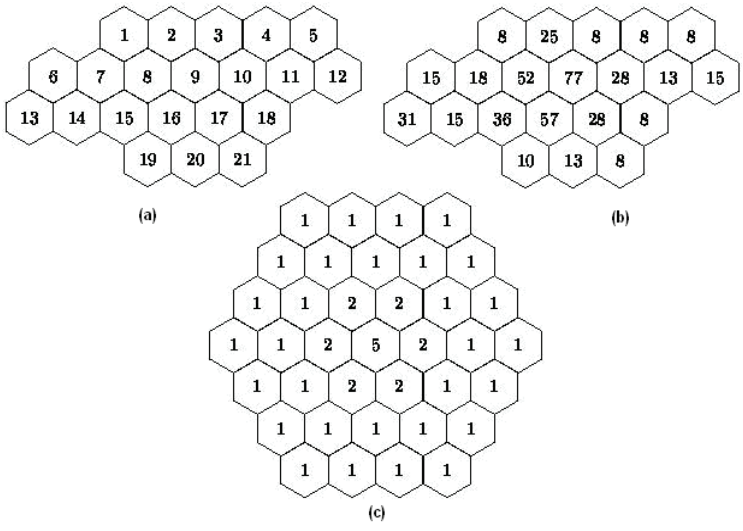

Figura 3.4. (a) Ejemplo de topología de antenas, (b) número de frecuencias a asignar a cada antena de la topología y (c) mínimas distancias de reutilización de frecuencias para evitar las interferencias en la red

Las instancias de Philadelphia se caracterizan por la topología de la red compuesta por 21 celdas hexagonales (Figura 3.4) que representan las celdas de la red de telecomunicaciones móviles que tuvo en su día la ciudad de Philadelphia. Cada antena necesita un número alto de frecuencias debido al número de comunicaciones que deben ser asignadas, pero como se ha explicado anteriormente, hay algunas restricciones que deben ser satisfechas para evitar insertar interferencias en la planificación (por ejemplo, antenas adyacentes no pueden tener asignadas las mismas frecuencias). Philadelphia representa un problema de planificación de frecuencias de tipo MS-FAP, por lo que el objetivo será obtener para cada instancia una solución libre de interferencias que minimice el rango de frecuencias utilizado (*span*) para dar cobertura a todas las comunicaciones de la red representada en la Figura 3.4.

De esta manera, en la Figura 3.4.a se puede ver un ejemplo de distribución de antenas en una red de celdas hexagonales que representa una topología de red como puede ser la del problema Philadelphia. Cada hexágono (celda) de la figura representa una antena que está etiquetada secuencialmente desde 1 hasta 21. En la Figura 3.4.b se puede observar el número de frecuencias que necesita cada antena para dar cobertura en su área (vector de requisitos). Este número no es constante, ya que cada celda necesita un número concreto de frecuencias para dar soporte a las comunicaciones que se producen en su zona. Por último, en la Figura 3.4.c hay un mapa de celdas donde el valor de cada una indica la mínima separación de distancia, en términos de número de frecuencias, en relación a la celda central para evitar las interferencias entre las celda que está siendo estudiada y las celdas que la rodean. Por ejemplo, el valor en la celda central de la Figura 3.4.c es 5, lo cual significa que en esa celda, la separación de las frecuencias que se asignen debe ser de 5 unidades.

El conjunto de problemas Philadelphia se resuelven en dos etapas. En primer lugar, es necesario encontrar una solución que no tenga interferencias en ninguna antena de la red, y a continuación, se deben encontrar diferentes soluciones que consigan que la distancia entre la frecuencia más alta y la más baja utilizada en la planificación sea mínima (problema MS-FAP). Es en esta segunda etapa donde entran en juego los algoritmos metaheurísticos, sin embargo, la resolución de problemas reales es mucho más compleja e

interesante que la resolución de instancias de prueba, por lo que esta tesis se ha centrado en este segundo tipo de enfoque.

3.2. La planificación de frecuencias en redes GSM

El problema de la asignación automática de frecuencias es una tarea muy importante para los operadores GSM (*Global System for Mobile communications*) actuales [2], ya que únicamente con una planificación óptima, que aproveche al máximo los escasos recursos en términos de ancho de banda con los que cuentan las operadoras actuales, es posible que se pueda mantener una comunicación de calidad entre los teléfonos móviles que utilizan una red GSM de telefonía.

El sistema GSM es la tecnología de más éxito para transmitir tanto voz como datos entre dispositivos móviles. La Figura 3.5 representa el esquema de la arquitectura que se utiliza en GSM. Como se puede observar, los dispositivos móviles se conectan de manera inalámbrica con las antenas de comunicación o BTS (*Base Station*). Cada antena, a su vez, está conectada con una estación controladora o BSC (*Base Station Controller*), que está unida de igual modo a un centro de conmutación de dispositivos móviles o MSC (*Mobile Switching Center*). Finalmente, estos centros están conectados a la red telefónica conmutada (o PTSN –*Public Swiched Telephone Network*). Por tanto, como está representado en la Figura 3.5, una red de comunicaciones está cubierta por una serie de antenas, o BTS, que se conectan a centros de control o BSC por áreas y éstos a su vez están unidos a centros de conmutación o MSC por donde se conectan a la red telefónica general.

En referencia al problema que se maneja en esta tesis, los dos elementos más importantes de la arquitectura GSM son los BTS y los TRX (transmisores-receptores), que se encuentran instalados en un número variable en cada BTS. Los TRXs están agrupados en sectores, existiendo normalmente de uno a tres sectores en cada antena, y su principal función es proporcionar la conversión entre el tráfico de datos digital con el que trabaja la red y las comunicaciones de

radio con las que se comunican el dispositivo móvil con la red GSM que le da servicio.

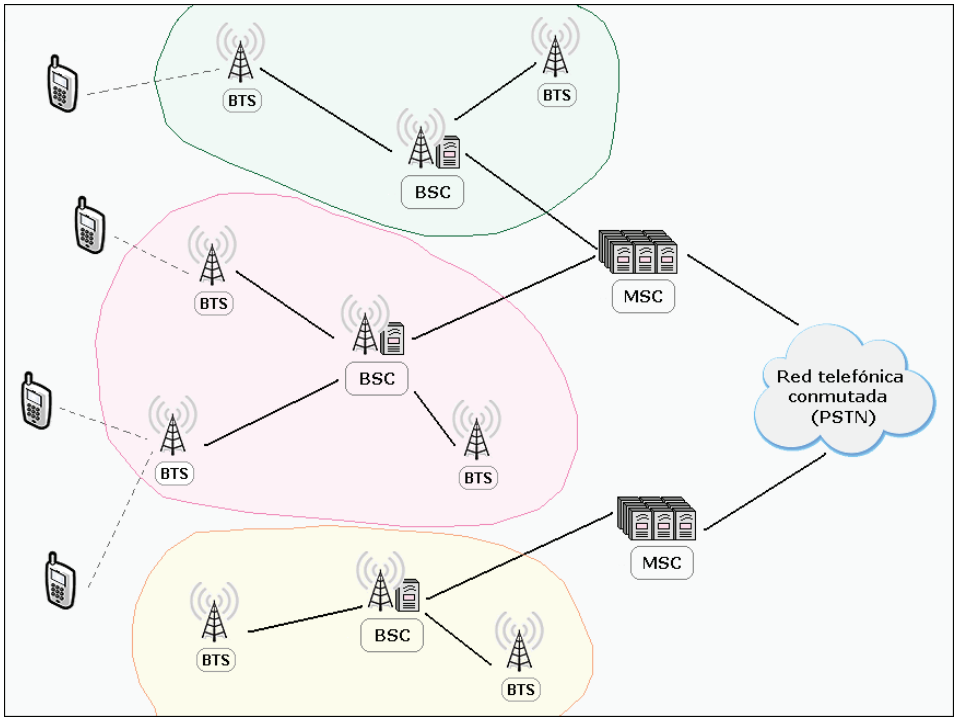

Figura 3.5. Esquema de la arquitectura GSM

Por tanto, un BTS es, en esencia, un conjunto de TRX (que está considerada la unidad de comunicación). En GSM, se utiliza el modo TDMA (*Time Division Multiple Access*) [147] mediante el cual un TRX puede ser compartido por hasta ocho usuarios, por lo que cada BTS puede utilizar múltiples frecuencias para dar cobertura a los usuarios de un determinado área. Sin embargo, el número de usuarios que deben soportar las redes GSM es enorme, por lo que se hace necesario replicar las frecuencias en diferentes TRX a fin de poder dar servicio a todos los usuarios de la red [2].

La planificación de frecuencias representa la última etapa en el diseño de una red GSM. El FAP consiste en la asignación de un canal, o una frecuencia, a todos los TRXs de una red [10]. El problema de optimización surge porque el número de frecuencias disponibles es normalmente muy reducido y, en consecuencia, las frecuencias deben ser reutilizadas por muchos TRXs. Sin embargo, el uso repetido de la misma frecuencia acarrea interferencias que

reducen la calidad de servicio (*QoS –Quality of Service*) hasta niveles que no son aceptables. De hecho, aunque la mayor fuente de interferencias se produce cuando se producen transmisiones sobre la misma frecuencia (interferencia co-canal), las transmisiones sobre frecuencias adyacentes también provocan interferencias debido a un solapamiento parcial del espectro, y han de ser tenidas en cuenta [24]. Este fenómeno aparece ilustrado en la Figura 3.2. El problema es que computar este nivel de interferencias es una tarea muy compleja que depende de varios factores, como son los canales disponibles, las señales de radio o las características del terreno. Sin embargo, es notorio que cuanto más precisa sea la forma de medir las interferencias, mejores y de mayor calidad podrán ser las planificaciones de frecuencias que se puedan generar para una determinada red. Existen una gran cantidad de métodos para cuantificar las interferencias que se producen en una determinada red. Estos métodos van desde aplicación de modelos teóricos hasta la toma directa de medidas, sin embargo, la forma más extendida de cuantificar las interferencias que se producen en una red GSM es utilizar la llamada matriz de interferencias [148]. Cada elemento de esta matriz es un par (i, j), que almacena los dos tipos de interferencias fundamentales que se producen entre todo par de TRXs de la red. De esta forma, el elemento primer elemento del par contiene la llamada interferencia co-canal, que representa la degradación de en la calidad de la red si las celdas i y j operan utilizando la misma frecuencia, mientras que el segundo elemento almacena la interferencia de canal adyacente (Figura 3.2) que se origina cuando dos TRXs utilizan canales adyacentes (por ejemplo, uno opera utilizando el canal f y el otro el canal f+1 o f–1).

En resumen, para realizar una planificación de frecuencias óptima es necesario disponer de una matriz de interferencias muy precisa y fidedigna, ya que el objetivo de cualquier algoritmo diseñado para resolver el FAP será minimizar la suma de todas las interferencias que están representadas en la matriz de interferencias. Además, en situaciones reales es posible que se produzcan otro tipo de complicaciones que hagan más compleja la planificación automática de frecuencias, como son el aumento del tráfico en determinadas áreas o los requerimientos de separación entre las distintas celdas [10].

3.3. Formulación matemática de un caso real

Los experimentos en los que se ha basado la mayor parte de la investigación llevada a cabo en esta tesis se han realizado utilizando instancias reales del problema FAP. Estas instancias han sido tomadas de la industria del sector, y para resolverlas se ha utilizado una formulación matemática que se ha adaptado expresamente al caso real con el que se ha trabajado. En la literatura hay multitud de aproximaciones matemáticas que se utilizan para modelar el problema de la planificación de frecuencias [7], si bien es bastante poco frecuente que estas aproximaciones se basen en información real. En cambio, el modelo matemático que se utiliza en esta tesis [24, 119] utiliza información procedente de la industria, y más concretamente, la matriz de interferencias objetivo, que no sólo almacena información acerca de las interferencias que se producen entre cada par de celdas de la red, sino también información de la distribución de probabilidad del ratio señal/ruido (C/I –*Carrier/Interference*) que se produce entre cada par de celdas de la red GSM con la que se trabaja.

En concreto, la formulación matemática que se ha utilizado (y que fue propuesta en [119]) es la siguiente: Sea $T = \{t_1, t_2,\ldots, t_n\}$ un conjunto de n trasmisores (TRX), y sea $F_i = \{f_{i1}, f_{i2},\ldots, f_{ik}\} \subset N$ un conjunto de frecuencias válidas que pueden ser asignadas a los transmisores $t_i \in T$, i=1,…, n. Nótese que el cardinal k de F_i no es necesariamente el mismo para todos los transmisores. Además se define $S = \{s_1, s_2,\ldots, s_m\}$ como un conjunto de sectores con cardinalidad m. Cada transmisor $t_i \in T$ es instalado en exactamente uno de los m sectores. De este punto en adelante se denotará al sector donde el trasmisor t_i sea instalado como $s(t_i) \in S$. Finalmente tendremos una matriz $M = \{(\mu_{ij}, \sigma_{ij})\}_{mxm}$, llamada matriz de interferencias. Los dos elementos μ_{ij} y σ_{ij} de la entrada de la matriz $M(i,j) = (\mu_{ij}, \sigma_{ij})$ son valores numéricos mayores o iguales a 0. De hecho μ_{ij} representa la media y σ_{ij} representa la desviación estándar de una distribución de probabilidades Gausiana describiendo el ratio de interferencias cuando el sector i y j operan en la misma frecuencia. Cuanto más alta se la media, menor será la interferencia y mejor será la calidad de comunicación. Nótese además

que la matriz de interferencias es definida a nivel de sector (celda), porque los trasmisores instalados en cada sector sirven la misma área.

Una solución válida para el problema se consigue mediante la asignación de cada transmisor de la red $t_i \in T$ a una frecuencia de F_i. A partir de ahora se denominará solución, o plan de frecuencias, como $p \in F_1 \times F_2 \times \ldots \times F_n$, donde $p(t_i) \in F_i$ es la frecuencia asignada al transmisor t_i. De esta forma, el objetivo es encontrar una solución p que minimice la siguiente función de coste:

$$C(p) = \sum_{t \in T} \sum_{u \in T, u \neq t} C_{sig}(p,t,u) \tag{3.1}$$

Para definir la función $C_{sig}(p, t, u)$ se definen los sectores donde los transmisores t y u están instalados como s_t y s_u. De esta forma, $s_t = s(t)$ y $s_u = s(u)$ respectivamente. Además tendemos que $\mu_{s_t s_u}$ y $\sigma_{s_t s_u}$ son los dos elementos correspondientes a la entrada $M = (s_t, s_u)$ de la matriz de interferencias con respecto a los sectores s_t y s_u. Entonces $C_{sig}(p, t, u)$ es igual a la siguiente expresión:

$$\begin{cases} K & si\ s_t = s_u,\ |p(t) - p(u)| < 2 \\ C_{co}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & si\ s_t \neq s_u, \mu_{s_t s_u} > 0,\ |p(t) - p(u)| = 0 \\ C_{adj}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & si\ s_t \neq s_u, \mu_{s_t s_u} > 0,\ |p(t) - p(u)| = 1 \\ 0 & en\ otro\ caso \end{cases} \tag{3.2}$$

La ecuación 3.2 representa los tres tipos de coste asociados a las tres tipos de interferencias que se pueden encontrar entre cada par de TRX dentro de una planificación de frecuencias. $K >> 0$ es una constante muy grande definida por el diseñador de la red de manera que no se asignen las mismas o frecuencias adyacentes a transmisores que sirven el mismo área (es decir, instalados en el mismo sector). Además la función $C_{co}(\mu,\sigma)$ representa el coste debido a las interferencias co-canal (ecuación 3.3), mientras que $C_{adj}(\mu,\sigma)$ es el coste en caso de las interferencias de canal adyacente (ecuación 3.6).

$$C_{co}(\mu,\sigma) = 100(1.0 - Q(\frac{C_{SH} - \mu}{\sigma})) \quad (3.3)$$

Donde el término *Q(z)* es la integral de cola de una distribución de probabilidad Gausiana de media cero y varianza uno (ecuación 3.4), mientras que C_{SH} representa el umbral mínimo de calidad en la señal.

$$Q(z) = \int_{z}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx \quad (3.4)$$

La función *Q* se utiliza ampliamente en los sistemas de comunicación digitales porque caracteriza la probabilidad de error de las señales digitales [149]. Esto significa que $Q(C_{SH}-\mu / \sigma)$ es la probabilidad de que el ratio C/I (*carrier-to-interference*, señal/interferencia) [147] sea mayor que C_{SH} (umbral mínimo de calidad de la señal) y por consiguiente $C_{co}(\mu s_t s_u, \sigma s_t s_u)$ indica la probabilidad de que el ratio C/I en el área del sector s_t esté por debajo de la calidad debido a interferencias provocadas por el sector s_u. Es decir, si esta probabilidad es baja, el valor de C/I en el sector s_t no es probable que sea degradado por interferencias producidas por el sector s_u y su calidad de comunicaciones tendrá un rendimiento alto, y al contrario, una probabilidad alta (y en consecuencia mayor coste) causará que el valor de C/I en dicho sector s_t sea degradado por interferencias producidas por el sector s_u y las comunicaciones perderán mucha calidad. Sin embargo, la función *Q* debe ser obtenida numéricamente, ya que no hay método exacto para calcularla cuando ésta tiene ningún método ni fórmula para ser calculada directamente. Por tanto, se utiliza la función de error complementario, *E* (ecuación 3.5).

$$Q(z) = \frac{1}{2} E\left(\frac{z}{\sqrt{2}}\right) \quad (3.5)$$

El método utilizado para el cálculo del valor del término *E* aparece descrito en la referencia [150], donde aparece explicado un método numérico que permite dicho cálculo con un error menor que $1{,}2 * 10^{-7}$. Análogamente la función $C_{adj}(\mu s_t s_u, \sigma s_t s_u)$ se define como aparece expresado en la ecuación 3.6.

$$C_{adj}(\mu,\sigma) = 100(1.0 - Q(\frac{C_{SH} - C_{ACR} - \mu}{\sigma})) \\ = 100(1.0 - \frac{1}{2}E(\frac{C_{SH} - C_{ACR} - \mu}{\sigma\sqrt{2}})) \quad (3.6)$$

La única diferencia entre las funciones C_{co} (ecuación 3.3) y C_{adj} (ecuación 3.6) es la inclusión de la constante $c_{ACR}>0$ (*adjacent channel rejection*) en la definición de la función C_{adj}. Esta constante de especificación hardware indica la capacidad de los receptores de recomponer señales que llegan distorsionadas por un canal adyacente. Nótese que el efecto de la constante c_{ACR} es que $C_{adj}(\mu, \sigma) < C_{co}(\mu, \sigma)$. Esto tiene sentido ya que utilizar frecuencias adyacentes no provoca interferencias tan fuertes como utilizar la misma frecuencia.

Finalmente, queremos remarcar que el modelo matemático que se ha expuesto en este capítulo, y que originalmente fue propuesto por Luna *et al*. en [119], incorpora la definición de una matriz de interferencias muy precisa que ha sido importada con datos reales tomados en la industria, y no automáticamente generados por ordenador. Esta información permite generar planificaciones de frecuencia de muy alta calidad, donde se tienen además en cuenta predicciones acerca de la calidad de servicio que podrá ofrecer la red, ya que tanto los datos de origen utilizados como las operaciones llevadas a cabo sobre ellos están basados en información real del modo en que se opera en una red GSM para realiza las planificaciones de frecuencias.

3.4. Resumen

En este capítulo se han explicado los fundamentos del problema que ha sido abordado en esta tesis. Se ha comenzado con una explicación genérica del mismo, indicando las variantes y tipos de problemas existentes. Asimismo, se ha hecho una mención especial al conjunto de instancias Philadelphia, ya que éstas representaron la primera aproximación que se realizó al problema FAP. Sin embargo, el resto del capítulo se ha centrado en la explicación del problema de

la asignación automática de frecuencias aplicado a redes de telecomunicaciones reales, ya que éste ha sido el caso en el que se ha centrado esta tesis. A este respecto, es importante subrayar que una red GSM da cobertura a sus usuarios a través de una serie de antenas, o BTSs, que se encuentran distribuidas en el área de terreno donde se quiere dar servicio. A su vez, en estas antenas se encuentran instalados una serie de sectores, que alojan un cierto número de TRXs, a los que se les debe asignar una frecuencia para hacer posible la comunicación entre la red y cualquier dispositivo móvil que la utilice. Por otra parte existen un conjunto de frecuencias disponibles para realizar esta operación y una matriz de interferencias que indicará las interferencias que se producen entre los distintos TRXs de la red.

Con todos los elementos mencionados, el problema de optimización surge debido a que hay muy pocas frecuencias para dar servicio a los millares de TRXs que conforman una red de telefonía real, por lo que es necesario que dichas redes dispongan de una planificación de frecuencias de calidad para aprovechar eficientemente el escaso espectro de radio del que disponen. A este respecto, es muy importante que la información a partir de la cual se generan dichas planificaciones provenga de datos de la máxima precisión, y que el modelo matemático utilizado se ajuste lo máximo posible a esos datos, ya que la calidad de las soluciones aportadas dependerá en gran medida de estos factores. En este capítulo se explica también el modelo matemático utilizado para abordar el caso real con el que se trabaja en esta tesis.

Capítulo 4

Metodologías aplicadas en la resolución del problema

En este capítulo se explican las estrategias que se han utilizado en esta tesis para dar solución al problema de optimización propuesto. El capítulo comienza justificando la utilización de estos métodos debido a la complejidad intrínseca del problema abordado. A continuación, se introducen, enmarcan y describen las estrategias metaheurísticas, haciendo hincapié en las que concretamente se han desarrollado en esta tesis. Además, debido al carácter estocástico de estos métodos, los resultados que se obtienen con su aplicación deben ser validados mediante análisis estadístico, por lo que en este capítulo se explicará también la forma en que se determina la calidad de los mismos. Finalmente, en esta tesis se resuelve un problema real de ingeniería utilizando instancias reales, por lo que en este capítulo se abordan diferentes esquemas paralelos que se han utilizado tanto para mejorar los resultados de las estrategias utilizadas como para reducir el tiempo de ejecución de los distintos experimentos realizados.

4.1. Problemas de optimización, complejidad computacional y estrategias de resolución

El problema que se aborda en esta tesis está incardinado dentro de lo que se denomina problema de optimización combinatorio. Este tipo de problemas están orientados, como su propio nombre indica, a la búsqueda de un valor óptimo,

que puede ser un valor óptimo mínimo (problema de minimización) o un valor óptimo máximo (problema de maximización). El caso concreto de la planificación automática de frecuencias es de búsqueda de un valor óptimo mínimo, ya que se trata de encontrar una planificación de frecuencias tal, que el coste asociado a todas las interferencias que se producen en la red sea mínimo. Un problema combinatorio de optimización (de búsqueda del valor mínimo) se puede definir de la siguiente manera [151, 152]:

Sea el par (S, f), donde S es un conjunto finito de soluciones candidatas y f es una función tal que $f: S \rightarrow R$ asigna a cada $s \in S$ un valor $f(s)$. A $f(s)$ también se le denomina valor de la función objetivo. El objetivo de un problema de optimización combinatorio es encontrar una solución $s \in S$ con un valor en su función objetivo mínimo, es decir, $f(s_{opt}) \leq f(s') \ \forall \ s' \in S$. Finalmente, s_{opt} es la solución óptima global de (S, f) y S_{opt} es el conjunto de todas las soluciones óptimas globales.

Lo habitual es que un problema de optimización combinatorio posea varios parámetros o variables que contengan valores indeterminados. Una instancia del problema se refiere al problema con todos sus parámetros o variables determinados. A la función objetivo se la denomina habitualmente función de coste o función de *fitness*, y a los valores de estas funciones, coste o *fitness*. Además, se asume que $f(s) \geq 0, \ \forall \ s \in S_k$.

Teóricamente, se puede abordar un problema de optimización combinatorio utilizando tres métodos:

- Los basados en búsquedas: Dada una instancia (S, f), se trata de encontrar una solución óptima tal que $s_{opt} \in S_{opt}$.

- Los basados en evaluaciones: Dada una instancia (S, f), se trata de encontrar el valor óptimo para la función objetivo, es decir: $f(s_{opt})$.

- Los basados en decisiones: Dada una instancia (S, f), y un valor de umbral L, se trata de decidir si la solución s es factible o no, es decir, encontrar una $s \in S$ tal que $f(s) \leq L$.

Los métodos más utilizados son los basados en búsquedas. Desde este punto de vista, se denomina a *S* el espacio de búsqueda de soluciones. Dado que se considera que *S* es finito, se asume que una instancia (*S*, *f*) puede resolverse enumerando el conjunto completo de soluciones y seleccionando entre éstas aquella con el valor de coste mínimo. Sin embargo, esta aproximación no es factible para muchos problemas debido a que el tamaño del espacio de búsqueda, denotado como |*S*|, crece de manera exponencial con el tamaño de las instancias.

El objetivo de cualquier método de resolución de problemas combinatorios es dar soluciones de la mayor calidad posible de manera tan eficiente y rápida como sean capaces. Un criterio extendido para clasificar este tipo de problemas es el tiempo que tardan en encontrar una solución válida algoritmos bien conocidos en la literatura. Estos aspectos son tratados por la teoría de la complejidad computacional, cuyo objetivo principal consiste en clasificar los problemas de acuerdo con la dificultad que conlleva su resolución. A este respecto, se dice que la complejidad de un problema viene determinada por la complejidad que conlleva resolver el peor escenario posible, o la instancia más difícil, de dicho problema. Los problemas NP-completos, o NP-duros, son los problemas dentro de la clase de complejidad NP más difíciles de resolver [123].

Existen dos clases de complejidad computacional. La clase de complejidad P, que está formada por los problemas que pueden ser resueltos en una máquina determinista en tiempo polinómico, lo que intuitivamente se corresponde con los problemas que se pueden resolver, aún en el peor de los casos, con un algoritmo que al ejecutarse en un ordenador conlleve un coste computacional polinómico. En otras palabras, la relación entre el tamaño del problema y el tiempo de ejecución que tarda el algoritmo en ejecutarse es polinómica. Por otro lado, están el conjunto de problemas que pueden ser resueltos por una máquina no determinista en tiempo polinómico. Este tipo de problemas se dice que pertenecen a la clase NP, y sus costes computacionales son factoriales o combinatorios. Por tanto, este tipo de problemas no pueden ser resueltos por una máquina determinista en un tiempo razonable.

No se tratará en esta tesis la teoría de la complejidad computacional, sin embargo, una conclusión que se extrae de manera casi directa de ella es que muchos problemas de optimización combinatorios no pueden ser resueltos

utilizando algoritmos exactos cuando se manejan instancias del problema de cierto tamaño y complejidad. En estos casos normalmente es necesario relajar la condición de encontrar soluciones óptimas por tratar de encontrar buenas soluciones en un tiempo que sea considerado razonable. Los algoritmos que se utilizan para resolver problemas de optimización combinatoria y que tratan de encontrar buenas soluciones en tiempos razonables se denominan algoritmos no-exactos, aproximados o heurísticos [123, 151].

Una pregunta natural que surge acerca de los algoritmos aproximados es cómo evaluar la calidad de las soluciones que se obtienen con éstos. En algunos casos, en los que se conoce la solución óptima del problema, basta con calcular el valor del error relativo respecto de dicha solución óptima, sin embargo, muchas veces el valor de la solución óptima no es conocido, como es el caso de esta investigación, por lo que hay que aplicar otros métodos de validación estadística y realizar comparaciones empíricas con resultados ya conocidos para determinar la calidad relativa de una nueva solución generada.

Clasificación de las técnicas de optimización

Las estrategias de resolución de problemas de optimización se pueden clasificar en dos categorías principales: estrategias exactas (o numerativas, exhaustivas, etc.) y estrategias no exactas, aproximadas o heurísticas [123]. Las estrategias o algoritmos exactos garantizan encontrar una solución óptima para cualquier instancia del problema en un tiempo que depende del tamaño de la instancia, pero que en cualquier caso debe ser finito. Debido a la complejidad inherente en muchos de los problemas de optimización combinatoria, que suele ser de tipo NP-completos, el inconveniente de este tipo de técnicas es que el tiempo y/o recursos computacionales que se necesitan para encontrar soluciones óptimas crece de manera exponencial con el tamaño de las instancias con las que se trabaja, por lo que el uso de estas técnicas puede ser inviable para problemas de cierto tamaño. Por tanto, este tipo de problemas exige de manera prácticamente obligatoria las estrategias no-exactas o aproximadas. Estos métodos sacrifican la garantía de encontrar una solución óptima a cambio de encontrar una solución

válida en un tiempo razonable. La solución hallada será habitualmente de alta calidad, y se obtendrá mucho más rápidamente que en el caso de los algoritmos exactos. La Figura 4.1 muestra una taxonomía general de las técnicas de resolución que se pueden aplicar para resolver problemas de optimización. Como puede observarse, existen también otros métodos de resolución, como pueden ser por ejemplo las redes neuronales o los algoritmos de razonamiento basados en reglas. No obstante, esta tesis se centra en el uso de técnicas metaheurísticas como estrategias de resolución, por lo que este capítulo se centrará en incardinar y describir las técnicas que se han empleado en nuestra investigación, más que en explicar de manera extensiva la diversidad de métodos que se pueden aplicar para la resolución de problemas de optimización, que sin duda son muchos.

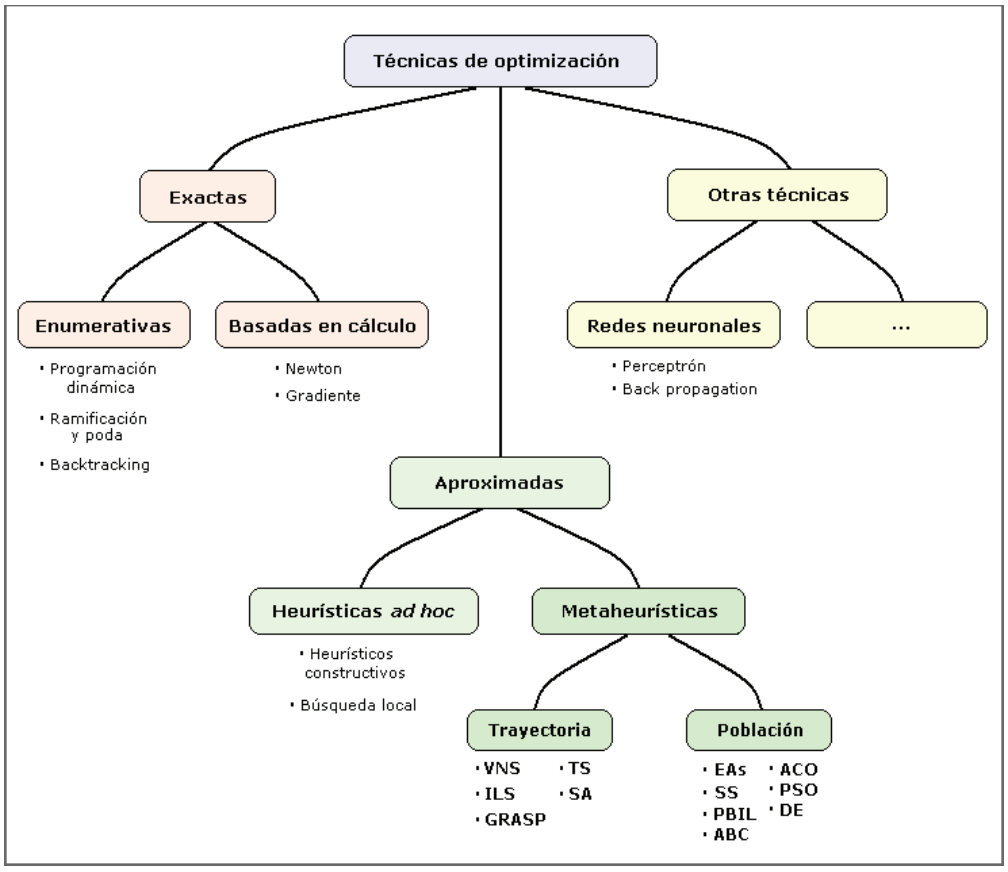

Figura 4.1. Clasificación de las estrategias de optimización

Dentro de los algoritmos aproximados destacan dos grandes categorías [24, 123]: los algoritmos heurísticos *ad hoc,* o construidos de manera específica para un problema concreto, y las estrategias metaheurísticas. Existen otros

métodos que se incluyen también dentro de conjunto de algoritmos aproximados, como son aquellas estrategias para las que se establece un límite inferior (en calidad, tiempo, etc.) para su terminación, sin embargo estas estrategias están íntimamente ligadas con el objetivo del problema con el que se trabaje (son dependientes del problema), lo cual limita su aplicabilidad. En cualquier caso, esta tesis se centrará en las estrategias metaheurísticas, ya que son las técnicas en las que se basa la investigación realizada. En cuanto a las estrategias heurísticas *ad hoc*, éstas pueden dividirse a su vez en algoritmos heurísticos constructivos y en estrategias de búsqueda local. El término "*ad hoc*" se refiere a que dichas heurísticas están adaptadas al problema de optimización al que son aplicadas, ya que tanto el éxito tanto de los algoritmos constructivos como de las técnicas de búsqueda local viene en gran parte determinado por el conocimiento que se tenga de los detalles del problema, y de lo eficiente que sea el diseño del algoritmo para el problema al que se aplican. Los algoritmos constructivos generan una solución desde cero mediante la incorporación de elementos hasta que la solución queda construida completamente. Este tipo de algoritmos aproximados son muy rápidos, aunque suelen devolver soluciones de baja calidad, y su éxito depende en mucha más medida que en las técnicas de búsqueda local del conocimiento que se tenga de los detalles problema. En muchos casos la forma más efectiva de obtener soluciones de alta calidad suelen ser las técnicas de búsqueda local o mejor aún, los algoritmos metaheurísticos. En contraposición con los algoritmos constructivos, las técnicas de búsqueda local parten de una solución inicial válida previamente generada, y en cada iteración tratan de conseguir una solución que mejore a la actual mediante la búsqueda de un óptimo local en el vecindario de dicha solución. El vecindario de una solución, s, es el conjunto de soluciones que se pueden construir a partir de variaciones de s aplicando un operador específico de modificación. Un óptimo local es una solución que mejore a cualquier otra de su vecindario. Por tanto, las estrategias de búsqueda local parten de una solución inicial (que muchas veces se genera aplicando algún algoritmo constructivo), y examinan su vecindario para quedarse con el mejor vecino. Este proceso se repite una serie de iteraciones hasta que o bien se encuentra un óptimo local o bien se satisface una condición de parada externa, ya que en muchos casos la exploración completa del vecindario es inabordable. Además, dependiendo del operador de perturbación que se utilice, la exploración del espacio de búsqueda puede verse alterada, simplificándose o complicándose.

En cualquier caso, la búsqueda local no es un método reciente para tratar problemas de optimización NP-completos, ya que se han encontrado trabajos publicados que datan de entre finales de los años cincuenta y principios de los sesenta [153, 154], donde este tipo de técnicas fueron aplicadas para resolver el problema del viajante de comercio. Las técnicas de búsqueda local han demostrado ser muy útiles cuando se aplican a instancias grandes de problemas de optimización, si bien los requerimientos tanto computacionales como temporales pueden suponer un gran hándicap en estos casos [155], ya que en muchas ocasiones la exploración del vecindario de soluciones para un determinado problema supone una tarea de costes prácticamente inabordables.

Sin embargo, en los años setenta surgió un nuevo tipo de algoritmos aproximados, cuya idea central era combinar diferentes métodos heurísticos a un nivel más alto de manera que se lograra una exploración del espacio de búsqueda eficiente y efectiva. Estas técnicas se denominan metaheurísticas. El término fue introducido por primera vez por Glover en la década de los ochenta [156] (antes de la aceptación del término, a estas técnicas se las denominaban heurísticas modernas). Esta clase de algoritmos aglutina una gran variedad de heurísticas, con diversas características, e inspiradas en varios paradigmas, como son los inspirados en la naturaleza, en la evolución de las especies, en el comportamiento de grupos de insectos, en la evolución de una única solución, de manera más estocástica o más avariciosa, etc. En la siguiente sección de este capítulo se detallarán las propiedades más importantes de este tipo de métodos, así como las características más destacadas de las técnicas metaheurísticas desarrolladas en el contexto de esta tesis doctoral para resolver el problema de la planificación automática de frecuencias.

4.2. Técnicas metaheurísticas

Una de las desventajas más importantes de las técnicas de búsqueda local es que debido a que buscan mejorar la solución con la que trabajan en un entorno cercano a la misma, pueden estancarse en lo que se llama un mínimo local. Una posible solución a esta circunstancia consiste en reiniciar la búsqueda utilizando

una nueva solución generada de manera aleatoria, o avariciosa, hasta que una condición de parada establecida se cumpla. Sin embargo, aunque esta estrategia puede conseguir buenos resultados para instancias pequeñas, pierde efectividad para problemas reales con instancias de grandes dimensiones, ya que el número de mínimos locales, o de máximos locales, crece de manera exponencial con el tamaño de las instancias del problema.

Desde finales de la década de los setenta han ido surgiendo una serie de esquemas heurísticos de propósito general con el objetivo de diseñar diferentes métodos genéricos que pudieran ser aplicables a un amplio rango de problemas de optimización combinatorio. A este conjunto de algoritmos se les denominó metaheurísticas, e incluye técnicas tan variadas como los algoritmos evolutivos, la búsqueda tabú, la búsqueda dispersa, la evolución diferencial o la búsqueda local iterada [123]. A este respecto, existen numerosos trabajos de gran calado sobre estas técnicas [11, 13, 123] que detallan las características fundamentales de estas estrategias, y que se pueden resumir en los siguientes cuatro puntos:

- Las metaheurísticas son algoritmos no exactos y por lo general no deterministas.

- El objetivo de las estrategias metaheurísticas es encontrar soluciones de alta calidad (óptimas o casi óptimas) mediante la exploración eficiente del espacio de búsqueda de soluciones de un problema determinado.

- Todas las técnicas metaheurísticas poseen un esquema básico genérico bien definido, si bien también pueden incluir heurísticas específicas, que utilizando conocimiento del problema, pueden adaptarse a las necesidades de éste para conseguir soluciones de mayor calidad.

- La diferencia entre las distintas estrategias metaheurísticas estriba en la forma en la que realizan la exploración del espacio de búsqueda.

Resumiendo los puntos anteriores se puede concluir que una metaheurística es una estrategia de alto nivel que usa diferentes métodos para explorar el espacio de búsqueda de soluciones. Es una plantilla general que debe ser completada con datos específicos del problema y que debe encontrar un equilibrio entre la dispersión y la intensidad de la búsqueda que realiza. De esta

forma, debe evitar realizar búsquedas en zonas poco prometedoras del espacio de búsqueda, y también ser capaz de intensificar la búsqueda cuando se encuentren zonas que puedan conducir a la solución óptima. Dependiendo de la forma en que realicen la exploración, existen diferentes clasificaciones para las metaheurísticas [11, 123]. Una de las más extendidas consiste en diferenciar las estrategias que son una extensión de los métodos de búsqueda local (llamadas metaheurísticas de una única solución o basadas en trayectoria) y las estrategias que realizan un muestreo sesgado de dicho espacio de búsqueda utilizando para ello un conjunto de soluciones heterogéneas (metaheurísticas basadas en población). Las primeras realizan la búsqueda utilizando un único de avance, mientras que las basadas en población se caracterizan por disponer de varios puntos de avance para explorar el espacio de búsqueda. Ejemplos de metaheurísticas basadas en trayectoria son: la búsqueda local iterada (ILS), la búsqueda con vecindario variable (VNS) o la estrategia de enfriamiento simulado (SA). En cuanto a las metaheurísticas basadas en población, algunos ejemplos son: los algoritmos evolutivos (EAs), la búsqueda dispersa (SS) o el algoritmo basado en colonia de abejas (ABC).

Otras formas de clasificación [123] se centran en si la búsqueda se realiza utilizando una única trayectoria (como en el enfriamiento simulado o la búsqueda tabú –TS) o bien realiza saltos en la exploración del vecindario (como en la búsqueda de entorno variable –VNS o en los EAs); en si la metaheurística tiene memoria acerca de la búsqueda que ya ha realizado (TS, EAs, ILS), o si no tiene en cuenta la experiencia previa para decidir la dirección que tomará la búsqueda en el futuro (SA, o en el procedimiento de búsqueda avariciosa, aleatoria y adaptativa –GRASP); o en si el esquema general de la metaheurística es bio-inspirado (EAs, ABC, ACO, SA) o no (TS, ILS, GRASP). Sin embargo, la forma más extendida de clasificación de las metaheurísticas es diferenciar las que utilizan una única solución en la búsqueda del óptimo de las que utilizan un conjunto de soluciones, población, y que por tanto tienen varios puntos de búsqueda. En las siguientes subsecciones se llevará a cabo una explicación de las metaheurísticas más representativas atendiendo a este último patrón de clasificación.

Clasificación de las metaheurísticas

Como se ha indicado anteriormente, existen varios parámetros por los que se pueden clasificar las técnicas metaheurísticas, sin embargo, uno de los más extendidos consiste en diferenciar aquellas técnicas que utilizan una única solución para explorar el espacio de búsqueda (metaheurísticas basadas en trayectoria), de aquellas estrategias que utilizan una población entera de soluciones que servirán para establecer diferentes puntos de búsqueda y realizar una exploración más dispersa (a éstas se las llama metaheurísticas basadas en población). En la Figura 4.2 aparecen reflejadas las técnicas desarrolladas en esta tesis atendiendo a esta clasificación.

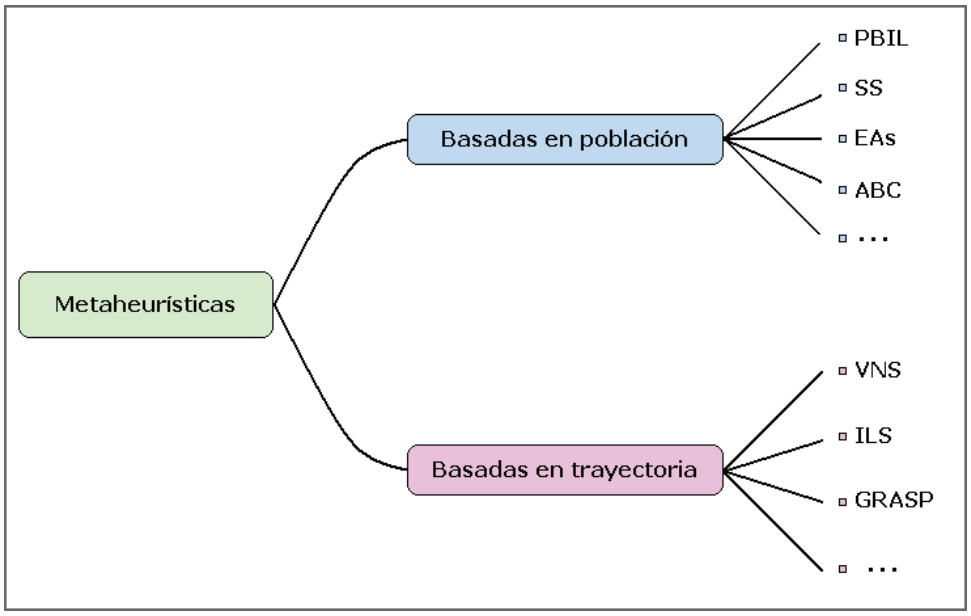

Figura 4.2. Clasificación de las técnicas metaheurísticas

Metaheurísticas basadas en trayectoria

Como ya se ha comentado, este tipo de estrategias tienen en común que utilizan una única solución en su exploración del espacio de búsqueda. Su forma de trabajar consiste en realizar cambios en dicha solución, de manera que se mejore la calidad de la misma en cada iteración mediante la exploración

sucesiva de su vecindario. Esta exploración va conformando una trayectoria en la búsqueda de la solución óptima, lo cual suele formar el nombre que las designa. Las metaheurísticas basadas en trayectoria son consideradas por muchos autores como extensiones de las estrategias de búsqueda local, pero con mecanismos específicos para añadir criterios de dispersión a tales búsquedas, de forma que se evite incurrir en mínimos locales. En esta sección se describen las metaheurísticas de este tipo más relevantes, haciendo especial hincapié en las que han sido desarrolladas para esta tesis.

Búsqueda de entorno variable (VNS)

La búsqueda de entorno variable o *Variable Neighbourhood Search* (VNS) [38, 39] fue propuesta por P. Hansen y N. Mladenovic a mediados de la década de los noventa. El algoritmo tiene como base una búsqueda local que cambia de entorno, o de vecindario, para reducir las posibilidades de que la trayectoria de evolución seguida por la solución sobre la que trabaja resulte en un mínimo local. El cambio de vecindario puede ser determinista, estocástico, o ambas cosas a la vez. Existen varias versiones del algoritmo, que se corresponden con diversas formas de realizar la búsqueda y de cambiar el entorno. Destacan la búsqueda de entorno variable descendente, la reducida, la básica y la general. Además, existen diferentes extensiones del algoritmo para dotarlo de algunas características adicionales, y hacer posible, por ejemplo, la resolución con éxito de problemas muy grandes. Destacan la búsqueda de entorno variable con descomposición (VNDS), la búsqueda de entorno variable sesgada (SVNS) y la búsqueda de entorno variable paralela (PVNS) [38, 39].

En cualquier caso, el algoritmo siempre comienza con la definición del entorno y la generación de una solución inicial. A continuación, cada iteración del algoritmo se divide en tres fases: primero, una fase de elección del vecino que se va a explorar, después, una etapa de búsqueda local con la que se trata de mejorar la solución y finalmente una etapa de actualización de la solución y cambio de entorno (o movimiento de vecindario). Este movimiento se puede producir o no, dependiendo de si se mejora la solución en la etapa de mejora. De esta manera, si la nueva solución es mejor que la actual, dicha nueva solución se convierte en la solución actual y se inicializa el contador de

vecindarios. En caso contrario, se repite el proceso pero utilizando el siguiente vecindario. Como se puede ver, el algoritmo tiene una parte de búsqueda intensiva, que se corresponde con su etapa de mejora (búsqueda local) y una etapa de diversificación, que se corresponde con su etapa de cambio de entorno.

Búsqueda local iterada (ILS)

La búsqueda local iterada o *Iterated Local Search* (ILS) [40, 41] basa su funcionamiento en la aplicación reiterada de perturbaciones y búsquedas locales sobre una solución dada. Fue introducida a finales de los noventa en la tesis doctoral de T. Stützle [157]. La entrada de este sencillo y rápido algoritmo de trayectoria es una solución previamente generada, a partir de la cual el algoritmo se divide en tres fases. En la primera, se realiza una perturbación o cambio en la solución actual. Dicha perturbación no debe ser ni muy pequeña, en cuyo caso puede que el algoritmo incurra en mínimos locales de los cuales no sea capaz de escapar, ni muy grande, en cuyo caso el algoritmo no podrá seguir una trayectoria definida, y se comportará como un algoritmo de búsqueda local con reinicio aleatorio. En la segunda etapa el algoritmo aplicará una operación de búsqueda local (LS –*Local Search*) sobre la solución actual. Finalmente ha de haber una operación de actualización que en base a un criterio de aceptación determine si la nueva solución generada a partir de la operación de LS sustituye a la solución previa. En muchos casos esta operación únicamente consiste en comparar la calidad de ambas soluciones, conservando aquella que proporciona una mejor solución al problema, sin embargo, en otras el esquema básico del de la metaheurística se mejora añadiendo alguna característica adicional, como puede ser la utilización de un histórico que determine el reinicio del algoritmo si tras un determinado número de iteraciones no se consigue mejorar la solución actual.

En cualquier caso, la operación de más relevancia en el algoritmo ILS es la búsqueda local. Esta operación debe ser tan efectiva y rápida como sea posible, ya que es la que hace mejorar la solución actual, y cuanto más efectiva sea, mejores serán los resultados que arroje el algoritmo ILS. En el caso más sencillo, este método de búsqueda local será una considerado una caja negra que será utilizada por el algoritmo en cada iteración, si bien es posible optimizar

la LS para realizar búsquedas más eficientes (por ejemplo, incluyendo pequeñas ejecuciones de otras heurísticas o incluso metaheurísticas –búsquedas tabú, enfriamiento simulado, etc.). El diseño de un buen algoritmo de búsqueda local que se adapte de la mejor forma posible al problema que se está tratando es muy importante, y muchas veces se trata de una operación no trivial, donde quizás haya que ponderar más la rapidez del algoritmo que su eficiencia, ya que sobretodo en problemas de grandes dimensiones, esta operación puede resultar muy costosa. Por esta razón, la elección de un buen algoritmo de perturbación resulta crítica para el buen funcionamiento del algoritmo, ya que debe ser una operación complementaria a la búsqueda local. A este respecto, es muy importante para la búsqueda de soluciones óptimas un buen balance entre la intensificación (aportado por la LS) y diversificación (aportado por las perturbaciones) que la metaheurística es capaz de conseguir, ya que en este equilibrio estará que el algoritmo no se estanque en mínimos locales ni pierda demasiado tiempo diversificando la búsqueda en exceso.

Procedimiento de búsqueda avariciosa, aleatoria y adaptativa (GRASP)

El procedimiento de búsqueda avariciosa, aleatoria y adaptativa o *Greedy Randomized Adaptive Search Procedure* (GRASP) [42, 43] fue introducido por T.A. Feo y M.G.C. Resende en 1995 [42]. El esquema del GRASP está dividido en tres fases: una primera etapa de construcción, donde se aplica una heurística constructiva; una segunda fase de mejora, donde se aplica una heurística de búsqueda local; y una fase de actualización. Por tanto GRASP se cataloga como una metaheurística constructiva (además de basada en trayectoria), ya que a diferencia de las técnicas anteriores, no parte de una solución ya generada, sino que la va construyendo con cada iteración del algoritmo. De acuerdo con el propio nombre del algoritmo, el proceso constructivo tiene tres características: avaricioso, aleatorio y adaptativo. Es avaricioso porque, en cada paso, la decisión de qué elemento incorporar a la solución se toma teniendo en cuenta qué elemento proporcionará mayor calidad a la solución en construcción teniendo en cuenta la función objetivo a optimizar. Para ello, en cada paso, se construye una lista con los elementos susceptibles de ser incorporados a la solución que se está construyendo. Esta lista se denomina lista de candidatos. A

cada uno de los elementos de dicha lista se le asocia un valor de bondad que determina lo adecuado que es ese elemento para ser añadido a la solución en construcción en ese momento determinado. Se dice que GRASP es adaptativo porque dado un elemento candidato, el valor de su índice de bondad puede cambiar de un paso a otro del algoritmo, es decir, que el interés que tiene ese elemento para la construcción de la solución puede variar en cada iteración del algoritmo dependiendo de las decisiones previamente tomadas. La tercera característica del proceso constructivo es la aleatoriedad, que se consigue seleccionando de forma aleatoria uno de los candidatos de mejor índice. Para esto es necesario conformar una lista de candidatos restringidos (RCL – *Restricted Candidate List*), que contendrá los elementos de la lista de candidatos que difieran menos del valor del mejor elemento de dicha lista. De esta forma, en cada iteración se elige aleatoriamente, pero de un conjunto de elementos que están seleccionados de manera avariciosa.

Aunque existen diferentes versiones del algoritmo GRASP dependiendo del método de construcción que se utilice, la metaheurística suele diseñarse "ad hoc" para cada problema, ya que dependerá en gran medida de las restricciones y características de éste. Sin embargo, esta es la principal desventaja del GRASP, ya que no todos los problemas permiten definir un índice que mida de forma efectiva el impacto de las sucesivas decisiones en la calidad de la solución finalmente generada.

Otras metaheurísticas basadas en trayectoria

Además de las metaheurísticas explicadas en los apartados anteriores, que se corresponden con las desarrolladas para abordar el problema de optimización con el que se ha trabajado en esta tesis, se ha querido incluir una breve descripción de otras técnicas basadas en trayectoria que por su relevancia merecen especial mención.

Búsqueda tabú (TS)

La búsqueda tabú o *Tabu Search* (TS) [156, 158, 159] tiene sus orígenes en F. Glover a finales de los años setenta [158], si bien el término no fue acuñado

hasta 1986 [156]. Esta estrategia es pionera en utilizar una memoria histórica para explotar las zonas más prometedoras del espacio de búsqueda, rechazando aquellas zonas ya exploradas o las que son susceptibles de contener mínimos locales. Esta forma de actuar puede considerarse hasta cierto punto determinista, de forma opuesta a otros algoritmos que confían casi exclusivamente en la aleatoriedad. Existen diversas formas de implementar la memoria histórica que utiliza la metaheurística, una de las cuales consiste en mantener una lista tabú donde se almacenan las soluciones visitadas más recientemente, de forma que se excluyan de los próximos movimientos. Sin embargo, esta forma resulta poco eficiente, ya que mantener una lista de soluciones en el caso de trabajar con problemas reales suele ser poco práctico. Es por esto que para casos reales la forma de diseñar la memoria consiste en almacenar los movimientos que llevaron a la metaheurística a generar esa solución, o bien los componentes principales la definen. En cualquier caso, los elementos contenidos en la lista tabú permiten filtrar el entorno de soluciones que serán elegibles en una determinada iteración del algoritmo. Para ampliar la información acerca de esta técnica, consúltese con la referencia [159], donde se explican en detalle los principios fundamentales y las distintas implementaciones que puede adoptar la TS.

Enfriamiento simulado (SA)

La técnica de enfriamiento simulado o *Simulated Annealing* (SA) es una de las metaheurísticas más clásicas, y posiblemente supone el primer algoritmo diseñado explícitamente para escapar de los mínimos locales. Su fundamentación se basa en el trabajo de Metrópolis *et al.* de 1953 [160] en el campo de la termodinámica estadística. De hecho, la idea del enfriamiento simulado es justamente simular el proceso de enfriamiento del metal o del cristal. A principio de la década de los ochenta, publicaciones independientes de Kirkpatrick et al. [161], y Cerny [162] mostraron cómo este proceso podría ser aplicado a problemas de optimización, asociando conceptos clave del proceso original de simulación con elementos de optimización combinatoria. Para evitar quedar atrapado en un mínimo local, esta metaheurística permite elegir con cierta probabilidad, T, una solución cuyo valor de *fitness* sea peor que el de la solución actual. T es un parámetro de control denominado generalmente temperatura, siguiendo la analogía con el proceso de enfriamiento físico. De esta

manera, cualquier implementación de búsqueda local puede convertirse en un proceso de SA al elegir elementos del entorno de manera aleatoria, aceptar automáticamente todos los movimientos hacia una mejor solución (es decir, el valor de la función de *fitness* es menor, en caso de minimización), y sólo aceptar los movimientos hacia una peor solución de acuerdo con una probabilidad que depende del valor de *T* (parámetro a configurar para cada problema) y de la diferencia de calidad (el valor del *fitness*) entre ambas soluciones. Por tanto, una solución que suponga una diferencia de calidad *t* en la función de coste, se aceptará con una probabilidad determinada por *T*.

Metaheurísticas basadas en población

Las metaheurísticas basadas en población se caracterizan porque en la exploración del espacio de búsqueda que realizan en cada iteración utilizan un conjunto de soluciones, una población de individuos. Por tanto, se diferencian de los métodos basados en trayectoria, que únicamente manejan una solución en cada iteración, en que la búsqueda es más dispersa y exhaustiva, lo cual reduce la posibilidad de quedarse atascados en mínimos locales e incrementa la probabilidad de encontrar el óptimo global. La contrapartida es que su evolución suele ser mucho más lenta, al tener que realizar muchas más operaciones por iteración, si bien son estrategias fácilmente paralelizables. A continuación se describen las metaheurísticas de este tipo que han sido desarrolladas para resolver el problema abordado en esta tesis.

Aprendizaje incremental basado en población (PBIL)

El aprendizaje incremental basado en población o *Population Based Incremental Learning* (PBIL) [36, 37] fue propuesto por S. Baluja a mediados de la década de los noventa [37]. PBIL combina estrategias evolutivas con un modelo probabilístico, por lo que se le considera en la bibliografía como un algoritmo de estimación de distribución [163]. Ese modelo probabilístico se basa en el aprendizaje competitivo, el cual está especialmente adaptado para la resolución de problemas de optimización. PBIL genera y mantiene un vector de

probabilidad que representa la tendencia de las mejores soluciones dentro del espacio de búsqueda que está siendo explorado. De esta manera, el vector de probabilidad representa la población que maneja el algoritmo, y será éste el que evolucione mediante operaciones de actualización utilizando la información genética del mejor individuo de la población además de diferentes parámetros probabilísticos. De esta forma, PBIL combina características de los algoritmos evolutivos, como son los conceptos de población, mutación, elitismo o evolución; con particularidades propias del aprendizaje competitivo, que es propio de las redes neuronales artificiales, y que hace que mediante el uso de reglas específicas (y parámetros como la tasa de aprendizaje o la probabilidad de mutación), dicho vector de probabilidad evolucione en cada iteración y genere soluciones cada vez de mayor calidad.

Además, dependiendo de la forma en que se haga evolucionar al vector de probabilidad, existirán diversas variantes del algoritmo. Así, por ejemplo, si no sólo se favorece que el vector de probabilidad evolucione hacia las mejores soluciones del espacio de búsqueda, sino que también se favorece que se aleje de las peores (mediante una tasa de aprendizaje negativo), la versión del algoritmo que se tendrá será la PBIL-LR-Negativa; si el vector de probabilidad se mueve por igual en base a las *M* mejores soluciones de la población, se estará hablando de la variante PBIL-M-Equitativa; mientras que si sólo cambian las zonas del vector de probabilidad donde las M mejores soluciones coincidan, estaremos enfrentándonos a la variante PBIL-M-Consenso. Para obtener más información sobre este algoritmo y sus diferentes variantes consúltense las referencias [36, 37].

Búsqueda dispersa (SS)

La búsqueda dispersa o *Scatter Search* (SS) [32-35] es una metaheurística que ha sido aplicada con éxito en un gran número de problemas de optimización. La primera descripción del método fue propuesta por F. Glover en 1977 [158]. El algoritmo se basa en mantener un conjunto de soluciones y realizar combinaciones con éstas, pero a diferencia del comportamiento habitual que define a la mayoría de las técnicas basadas en población, que trabajan en base a procesos estocásticos realizados sobre un conjunto de individuos relativamente

grande, SS funciona aplicando cambios estratégicos sobre un conjunto de soluciones que puede ser considerado bastante pequeño. A este conjunto de soluciones representativas de la población se le denomina conjunto de referencia (*RefSet*) y se caracteriza por contener dos tipos de soluciones. Por un lado aquellas que tienen mayor calidad y por otro aquellas que presentan mayor diversidad (las más distantes en el espacio de búsqueda).

La búsqueda dispersa consta de los siguientes cinco elementos principales: un generador de soluciones diversas, con el que se crearán un conjunto, *P*, bastante grande y heterogéneo de soluciones diversas del cual se extraerán los individuos para generar el *RefSet*; un conjunto de referencia, que se extraerá de forma equilibrada de *P* para aportar soluciones de calidad y soluciones diversas al conjunto, y que se irá actualizando y mejorando con las iteraciones del algoritmo; un generador de subconjuntos de soluciones, que se encargará de generar subconjuntos de dos o más individuos del *RefSet*; un método de combinación, con el que se mezclará el material genético de los individuos contenidos en los subconjuntos previamente generados y se actualizará el *RefSet*, bien dinámicamente (actualizando el conjunto de referencia cada vez que se genera una solución nueva) o bien estáticamente (manteniendo una lista con todas las soluciones generadas en una iteración y actualizando el *RefSet* cuando todas han sido generadas y evaluadas); y un método de mejora, que típicamente es un método de búsqueda local para mejorar las soluciones, tanto cuando son introducidas en el *RefSet* al comienzo del proceso, como las que se van generando fruto de la combinación de individuos.

Además, es frecuente que esta técnica se complemente con diversas mejoras de diseño que pueden aumentar la calidad de las soluciones que se obtienen. De esta forma, por ejemplo, se puede dotar a la búsqueda dispersa de un proceso de reinicio, de manera que se cree un nuevo conjunto de referencia que contenta por un lado las mejores soluciones que se encuentren en el *RefSet* actual, mientras que los restantes individuos sean generados mediante el método de generación de soluciones diversas. Para más información sobre las mejoras que pueden aplicarse a esta metaheurística, consúltense las referencias [32-35].

Algoritmos evolutivos (EAs)

Los algoritmos evolutivos o *Evolutionary Algorithms* (EAs) [31] son un conjunto de técnicas estocásticas que basan su comportamiento en ciertos procesos de la teoría de la evolución darwiniana. Todos ellos siguen un proceso iterativo durante una serie de lo que se denominan generaciones en este entorno, y operan sobre una población de soluciones, llamadas individuos en el contexto en el que se encuadran esta familia de algoritmos. La población inicial de individuos suele ser generada de manera aleatoria cuando comienza el proceso, si bien es posible utilizar algún tipo de heurística constructiva para conseguir individuos de mayor calidad de partida.

El esquema básico de los EAs está dividido en tres fases: selección de los mejores individuos, reproducción de éstos para producir la nueva generación de soluciones y reemplazo de la población actual, o de parte de ella, con la nueva generación de individuos. Este proceso es repetido hasta que se cumpla un cierto criterio de terminación, que suele ser un número de iteraciones o bien una condición temporal. Por tanto, los EAs tienen en común las siguientes características: En primer lugar, utilizan un método de selección sesgado en base a la calidad del individuo candidato. De esta manera, cuanto mejor es el valor de la función que evalúa la calidad del individuo, es más probable que éste sea seleccionado en el proceso evolutivo, y por lo tanto será más probable que su material genético pase a las siguientes generaciones de individuos. En segundo lugar, generan nuevos individuos a través de mecanismos de herencia de los individuos existentes en la población actual. Los descendientes son generados utilizando dos operadores estocásticos sobre la población actual de individuos, que en la mayoría de los casos son el cruce (o combinación) de individuos y la mutación. El cruce intercambia el material genético de generalmente dos individuos, mientras que la mutación cambia de manera normalmente aleatoria una pequeña parte del material genético de un individuo. Finalmente, la nueva población de soluciones sustituirá a la anterior, permitiendo, si se configura para el algoritmo una evolución que incluya elitismo, que los mejores individuos de la población anterior se mantengan en la nueva.

Los EAs tienen un origen disperso que se puede datar a finales de los años cincuenta [164-166]. Dichas estrategias han ido evolucionando y han dado lugar a diferentes técnicas evolutivas que hoy día son ampliamente utilizadas en la resolución de problemas de optimización. Entre estas técnicas destacan: la programación evolutiva (*Evolutionary Algorithm*, EA) [167], las estrategias de evolución (*Evolutionary Strategies*, ES) [168], los algoritmos genéticos (*Genetic Algorithm*, GA) [29] y la programación genética (*Genetic Programming*, GP) [169]. Aunque todas ellas tienen un esquema de alto nivel común, y elementos en los que coinciden, cada una incorpora sus detalles de diseño e implementación de las distintas operaciones. En esta tesis se ha desarrollado un algoritmo genético, que es el tipo de EA más comúnmente utilizado.

Algoritmo basado en colonia de abejas (ABC)

El algoritmo basado en colonia de abejas o *Artificial Bee Colony* (ABC) [44-46] basa su funcionamiento en la inteligencia de las abejas recolectoras de miel. Fue introducido por D. Karaboga en el año 2005 [46] y está especialmente diseñado para resolver problemas de optimización. La metaheurística incluye varios tipos de operaciones de búsqueda de acuerdo con los comportamientos que tienen los distintos tipos de abeja dentro de una colonia. En concreto, se distinguen tres clases de abejas: las observadoras (*onlooker bees*), las obreras (*employed bees*) y las exploradoras (*scout bees*). Todas ellas buscan fuentes de alimento (que siguiendo la metáfora utilizada son las soluciones al problema abordado) que sean de la mejor calidad posible, por lo que las fuentes deben ser evaluadas de acuerdo con su calidad. Por tanto, es necesario que el algoritmo defina una función de *fitness* que establezca un valor de calidad para cada solución que se examina. Las abejas observadoras realizan una búsqueda de fuentes de calidad dentro del espacio de soluciones en el que trabajan las abejas obreras, fijándose en la danza que realizan este tipo de abejas para ellas, o en otras palabras, examinando la calidad de las soluciones disponibles. Siguiendo el símil, las fuentes de calidad representan buenas soluciones para el problema de optimización que se maneja, por lo que el algoritmo realiza la búsqueda intensiva en las mejores zonas del espacio mediante las abejas obreras y las abejas observadoras. Además, para realizar una búsqueda lo más amplia

posible, el tercer tipo de abejas, las exploradoras, se envían cada cierto número de iteraciones para dotar de diversidad al proceso de búsqueda llevado a cabo por los otros dos tipos de abeja.

Por tanto, se puede decir que el algoritmo ABC va desplazando la población de abejas hacia las mejores soluciones del problema. Este proceso se realiza mediante mecanismos de búsqueda de entorno sobre las áreas del espacio de búsqueda que son más prometedoras, mientras que se tratan de evitar zonas donde se encuentran soluciones de peor calidad. El proceso comienza con una generación aleatoria de la población a la que seguirán tres etapas que se repiten en un bucle hasta que se cumpla la condición de terminación del algoritmo. La primera etapa consiste en la búsqueda en el vecindario de las soluciones almacenadas en la población actual llevada a cabo por las abejas obreras. A continuación las abejas observadoras generarán nuevas soluciones a partir de las mejores candidatas en las que trabajan las abejas obreras. Finalmente, las abejas exploradoras generarán nuevas soluciones, de manera aleatoria, las cuales reemplazarán a las peores candidatas de la población.

Otras metaheurísticas basadas en población

Las metaheurísticas descritas en los apartados anteriores se corresponden con las técnicas basadas en población desarrolladas para resolver el problema de optimización abordado en esta tesis. Sin embargo, se ha considerado relevante introducir una breve descripción de otras técnicas basadas en población que por su relevancia bibliográfica merecen especial atención.

Algoritmo basado en colonia de hormigas (ACO)

El algoritmo basado en colonia de hormigas o *Ant Colony Optimization* (ACO) [170, 171] fue introducido por M. Dorigo en 1992 [170]. Es un método probabilístico pensado para resolver problemas combinatorios que está inspirado en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento. Una vez encontrada una fuente de alimento, las hormigas depositan una hormona llamada feromona a lo largo de su camino de regreso a la colonia. Si otras hormigas encuentran este rastro,

lo más probable es que sigan el camino marcado para depositar el alimento en la colonia. Con el paso del tiempo el rastro de la feromona comienza a evaporarse y se reduce su fuerza atractiva. Las hormigas que siguen el rastro aumentan la cantidad de la feromona, con lo que el rastro es más fuerte y persistente en el tiempo. Por tanto, cuantas más hormigas recorran ese camino, más intenso será el olor de la feromona, lo que estimula a más hormigas a seguir esa trayectoria. La comunicación indirecta entre hormigas por medio de feromonas permite que éstas encuentren el camino más corto entre la colonia y la fuente de alimento. El algoritmo ACO intenta simular este comportamiento para resolver problemas de optimización. La técnica se basa en dos pasos principales: construcción de una solución basada en el comportamiento de una hormiga y actualización de los rastros de feromona artificiales. Para obtener más información sobre esta metaheurística, consúltense las referencias [170, 171].

Algoritmos basados en cúmulos de partículas (PSO)

Los algoritmos basados en cúmulos de partículas o *Particle Swarm Optimization* (PSO) [172, 173] son un conjunto de metaheurística inspirada en el comportamiento social de ciertos seres vivos que interactúan en grandes grupos, como son las bandadas de aves o los bancos de peces. La idea para su creación fue propuesta por J. Kennedy y R. Eberhart a mediados de los noventa [172]. Estas técnicas comparten muchas ideas extraídas de los algoritmos evolutivos, como por ejemplo, la inicialización aleatoria de la población al comienzo del proceso o la búsqueda de soluciones óptimas a través de la mejora generacional de la población. No obstante, los PSO no tienen operadores evolutivos como pueden ser el cruce o la mutación. Un algoritmo PSO consiste en un proceso iterativo y estocástico que opera sobre un cúmulo de partículas. La posición de cada partícula representa una solución potencial, y éstas se van moviendo por el espacio de búsqueda tratando de encontrar soluciones óptimas para el problema que se está resolviendo. Cada partícula guarda las coordenadas de su localización en el espacio de búsqueda, así como las de la mejor solución encontrada hasta la fecha, tanto de manera global, como en su entorno, de acuerdo con un valor de *fitness*. Si suponemos el caso concreto de una bandada de aves que buscan una fuente de alimento, cada solución (partícula) es un ave en el espacio de búsqueda que está en continuo movimiento. El cúmulo de partículas (*swarm*) es un sistema multiagente, es decir, las partículas son

agentes simples que se mueven por el espacio de búsqueda y que guardan a la vez que comunican la mejor solución que han encontrado. El movimiento de las partículas está guiado por las partículas de mayor calidad en el momento actual. De esta manera, en cada iteración se modifica de manera aleatoria la velocidad (o aceleración) con la que cada partícula se dirige hacia las mejores soluciones encontradas hasta ese momento. Una vez realizado el movimiento de las partículas, se recalcula el valor de fitness y se actualizan los valores de las mejores posiciones encontradas hasta entonces. El vector velocidad de cada partícula se actualiza en cada iteración utilizando la velocidad anterior, un componente cognitivo y un componente social. Dependiendo de los diversos valores que puedan adoptar estos parámetros se hablará de un tipo u otro de algoritmo basado en cúmulo de partículas. Para obtener más información acerca de los detalles y variantes de los PSO, consúltense las referencias [172, 173].

Evolución diferencial (DE)

La evolución diferencial o *Differential Evolution* (DE) [174, 175] es una metaheurística evolutiva propuesta por K. Price y R. Storn a mediados de los años noventa [174] diseñada para resolver problemas de optimización. El algoritmo comienza con la inicialización aleatoria de una población de soluciones, las cuales suponen el comienzo de la búsqueda. A continuación el algoritmo se divide en tres etapas que serán repetidas una serie de iteraciones hasta que la condición de terminación se cumpla. La primera fase consiste en una mutación con la que se modifican las soluciones de la población actual a partir de ciertos individuos elegidos al azar y de una cierta probabilidad de mutación. La siguiente es una recombinación donde se mezcla de manera aleatoria el material genético de las soluciones originales con el material genético de los individuos que fueron mutados en la fase anterior. Finalmente hay una etapa de selección donde se realiza una comparación por calidad entre el individuo original y el nuevo individuo mutado y recombinado. Aquel que sea mejor será el candidato para formar la población de la siguiente generación. Para más información sobre esta técnica, consúltense las referencias [174, 175].

4.3. Evaluación estadística de las metaheurísticas

Como se ha expuesto en las secciones anteriores de esta tesis, las metaheurísticas son estrategias aproximadas no deterministas, por lo que ejecutar diferentes veces la misma estrategia sobre un mismo problema no garantiza encontrar la misma solución. Dado un problema de optimización combinatoria, el objetivo de toda metaheurística consiste en encontrar soluciones de alta calidad para ese problema en un tiempo razonable. Los parámetros "soluciones de alta calidad" y "tiempo razonable" resultan a veces difíciles de establecer, ya que en muchas ocasiones no se conoce cuál es la solución óptima del problema, y dependiendo de los requerimientos de éste, el tiempo será un parámetro más o menos crítico.

De cualquier forma, el tiempo que se permitirá a una metaheurística estar buscando una solución óptima de un problema concreto deberá estar acotado, e incluso puede ser el parámetro que fuerce la terminación de la ejecución del algoritmo, sobre todo cuando estas estrategias se aplican a problemas de grandes dimensiones, como es el caso de la presente tesis.

Por otro lado, derivado del hecho de trabajar con metaheurísticas y de no conocer en muchas ocasiones cuál será la solución óptima para el problema concreto con el que se trabaja, se hace necesario realizar comparaciones empíricas con las que se permite evaluar la calidad de las soluciones aportadas por estas técnicas. En este sentido, los resultados deben ser validados con un número mínimo de ejecuciones independientes, las cuales, además, deben ser analizadas estadísticamente. Así pues, se recomienda realizar, al menos, 30 ejecuciones independientes con cada algoritmo metaheurístico que se esté evaluando, tras lo cual debe realizarse un análisis estadístico global para poder afirmar que los datos obtenidos son estadísticamente significativos y no fruto de variaciones aleatorias [176-178]. Tras esta valoración, los resultados deben ser comparados empíricamente con otros datos obtenidos mediante otras técnicas, o por otros autores, de manera que su calidad pueda ser contrastada mediante examen comparativo.

Desde el punto de vista estadístico, los resultados que se obtienen con las ejecuciones independientes que se hacen con cada estrategia pueden ser consideradas como muestras de una distribución de probabilidad. Para dotar de validez científica a dichos datos, se puede realizar el análisis que se describe en la Figura 4.3.

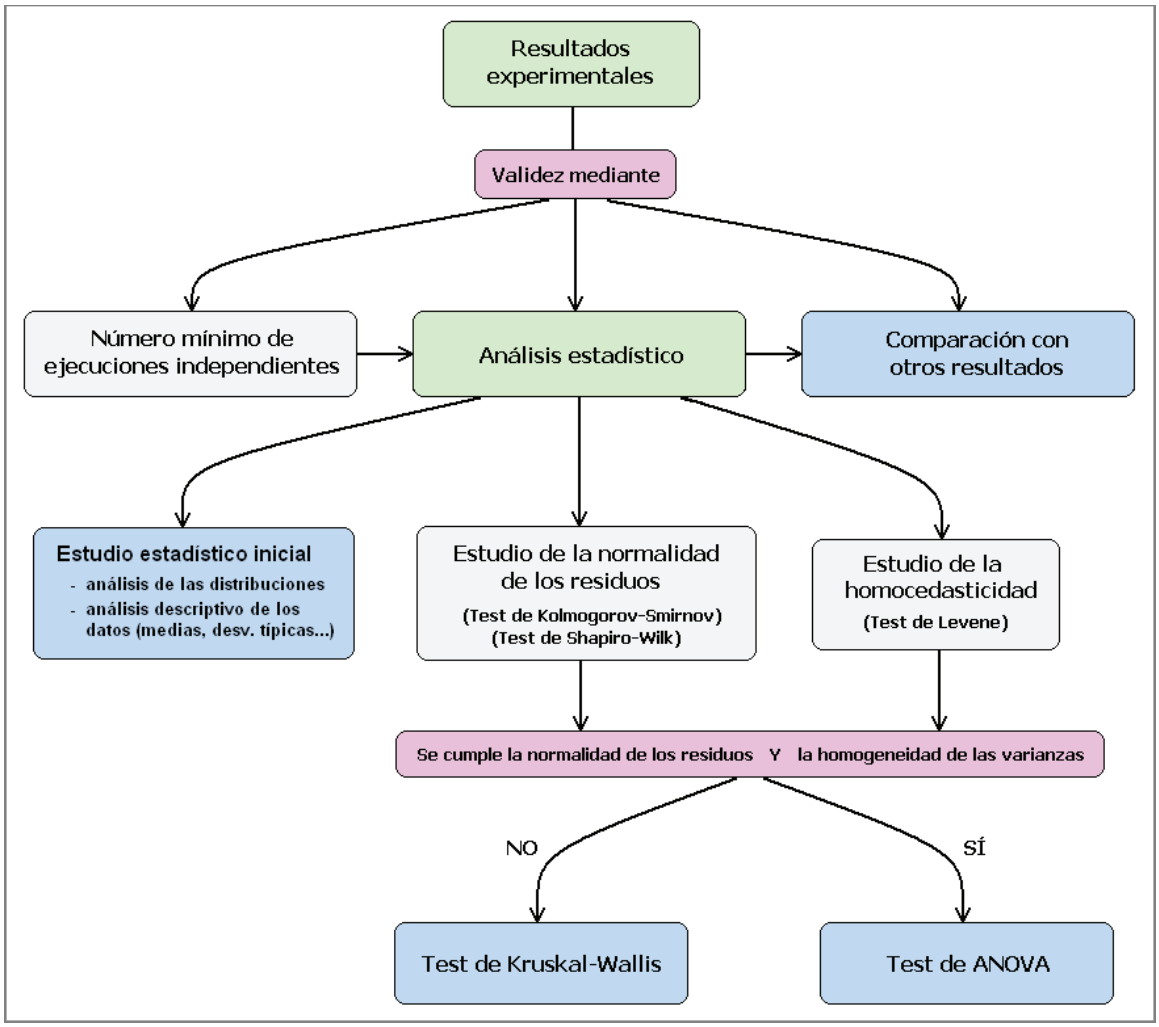

Figura 4.3. Resumen gráfico del análisis estadístico aplicado a los datos

Sin embargo, antes de utilizar cualquier técnica estadística, deben tenerse en cuenta las condiciones de aplicación de la misma, ya que antes de utilizar un método determinado hay que comprobar que se cumplan los requisitos de ese método, debiendo aplicar un análisis alternativo en caso de no cumplirse. En este trabajo se ha estudiado, en primer lugar, la distribución de los datos mediante un diagrama de dispersión. De esta forma se comprueba que los datos provenientes de los distintos experimentos presentan una relativamente baja

dispersión. A continuación se ha realizado un estudio descriptivo de los datos examinando las medias y desviaciones típicas, poniendo especial atención en los resultados atípicos obtenidos. Finalmente se ha realizado un análisis global de los datos, bien mediante un análisis de la varianza (test de ANOVA), o caso de no poder aplicarse ANOVA (Figura 4.3), mediante un estudio no paramétrico (test de Kruskal-Wallis). Lógicamente el análisis descrito en este capítulo no es único, pudiéndose aplicar muchos otros métodos [176-178], pero el que aquí se propone resulta adecuado tanto por la naturaleza de los datos como por el objetivo que se persigue con dicho análisis, que no es otro que el de verificar que los datos son relevantes estadísticamente y no fruto de la casualidad.

Para poder aplicar el test de ANOVA de una vía se deben cumplir los siguientes 4 puntos:

1. La variable dependiente debe medirse al menos a nivel de intervalo, lo cual significa que debe ser continua, y en el caso de todos nuestros experimentos es satisface esta condición, ya que nuestros resultados están dados en función del coste asociado a las soluciones que generan cada una de las estrategias que aplicamos, el cual se mide de manera continua.

2. Las observaciones a las que se somete este estudio deben ser independientes. Esta condición también se cumple, ya que todos nuestros experimentos son fruto de, al menos, 30 ejecuciones independientes.

3. Las varianzas de las distintas subpoblaciones deben ser constantes. A esta propiedad se le llama homocedasticidad (u homogeneidad de las varianzas), y se puede investigar mediante el test de Levene.

4. Los datos en cada muestra deben proceder de una subpoblación normal. Esta suposición se investiga en base a los residuos, que son las diferencias entre las observaciones en una muestra y la media de tal muestra. El estudio de la normalidad puede llevarse a cabo mediante varias técnicas, como la observación de histogramas que pueden generarse con los residuos, o la generación de gráficos Q-Q normales de residuos para los valores de coste, si bien en este trabajo se han llevado

a cabo mediante las pruebas de Kolmogorov-Smirnov (límite inferior para el grado de significación) y Shapiro-Wilk (valor más exacto del grado de significación).

Por tanto, una vez verificados los puntos anteriormente descritos, se puede realizar el test de ANOVA para investigar si los costes promedios asociados con los distintos algoritmos metaheurísticos son significativamente diferentes o no. En caso de no poder aplicar ANOVA, bien porque la distribución de los residuos no sea normal o bien porque las varianzas no sean constantes, se puede aplicar el test de Kruskal-Wallis para estudiar la igualdad de las medianas (de las subpoblaciones). También sería posible realizar otros test no paramétricos, como el de Wilcoxon o el de Wetch [177]. El test de Kruskal-Wallis es un contraste no paramétrico, y por tanto menos restrictivo que la de ANOVA. Al tratarse de un método estándar, el análisis de la varianza (ANOVA) es una prueba más potente que cualquier método no paramétrico, si bien este tipo de pruebas, y en concreto la de Kruskal-Wallis, se utilizan cuando no se cumplen las condiciones anteriormente mencionadas para aplicar la prueba estándar, ya que sus requisitos de aplicación son menores.

Además, es posible investigar aspectos adicionales de los datos. Por ejemplo, en general resulta interesante estudiar los valores atípicos, mediante por ejemplo, diagrama de cajas de los residuos. De esta manera, se identifican los valores extremos de cada experimento. Con esta información, se podrían eliminar dichos casos extremos del estudio, y repetir el análisis estadístico anteriormente mencionado, ya que dichos valores atípicos pueden no ser representativos respecto al conjunto de muestras que se están estudiando.

Finalmente, queremos indicar que el hecho de que los datos no superaran pruebas de diferenciación significativa de la media o de la mediana no eliminaría toda relevancia estadística de los resultados obtenidos de manera empírica. No satisfacer estas propiedades, pese a que son importantes, no elimina todo su valor, ya que los datos obtenidos pueden ser relevantes debido a que sigan distribuciones distintas, o porque cubran otros rasgos particulares que los hagan interesantes. Por tanto, ya sea por la significación de los valores medios, por la baja dispersión que presenten, o por su singularidad, todos los resultados presentados en esta tesis son relevantes estadísticamente (ver Apéndice I).

4.4. Estrategias paralelas

La computación paralela tiene por objetivo la ejecución eficiente de aplicaciones intensivas bien en datos o en cálculos. Este hecho se ajusta perfectamente al problema que se plantea en esta tesis, el problema de la asignación de frecuencias (FAP), y a las técnicas de resolución que se utilizan (metaheurísticas), ya que el problema FAP tiene unas necesidades computacionales altas (es un problema NP-Completo del que se han utilizado además instancias reales de gran tamaño) y las búsquedas heurísticas y metaheurísticas conllevan habitualmente una enorme carga computacional tanto en los datos que se deben almacenar como en sus cálculos (la evaluación de la función de *fitness* suele ser costosa para problemas de grandes dimensiones, manejan una población de individuos durante un elevado número de generaciones, etc.). Por tanto, el uso de técnicas de computación paralela ayuda a mejorar tanto los resultados que se consiguen con estas estrategias como el tiempo en el que estas técnicas arrojan soluciones de calidad.

Existen una gran cantidad de trabajos publicados donde se explican los fundamentos de las metaheurísticas paralelas y de las estrategias de paralelismo que se pueden aplicar a estas técnicas [47-55]. En cualquier caso, de manera general, las técnicas de paralelismo se pueden clasificar en dos categorías. Por un lado, las estrategias que utilizan paralelismo de grano fino, donde se modifica el propio algoritmo para que se ejecute de manera paralela. Las metaheurísticas se adaptan perfectamente a este paradigma, ya que en el caso de las metaheurísticas basadas en población se podría aplicar, por ejemplo, un modelo de islas donde se ejecutaran de manera paralela distintas porciones de la población. En cuanto a las metaheurísticas basadas en trayectoria, se podría aplicar, por ejemplo, un modelo de ejecuciones múltiples con el que se consigue que varias instancias del algoritmo se ejecuten de manera paralela coordinándose cada cierto tiempo.

El paralelismo de grano fino es el que da origen a la mayoría de las metaheurísticas paralelas, cuyo objetivo suele ser el de mejorar los resultados

(en calidad, tiempo o en ambos parámetros) que se consiguen con las versiones secuenciales. En contraposición y complementación con esta forma de realizar paralelismo se encuentra el llamado paralelismo de grano grueso, o paralelismo externo, que consiste en realizar de manera paralela múltiples ejecuciones del algoritmo secuencial, de forma que se consiguen más resultados en menos tiempo. Este tipo de paralelismo es propio de los sistemas de alta productividad, como los sistemas grid. La Figura 4.4 ilustra los dos tipos de entornos paralelos que se pueden encontrar.

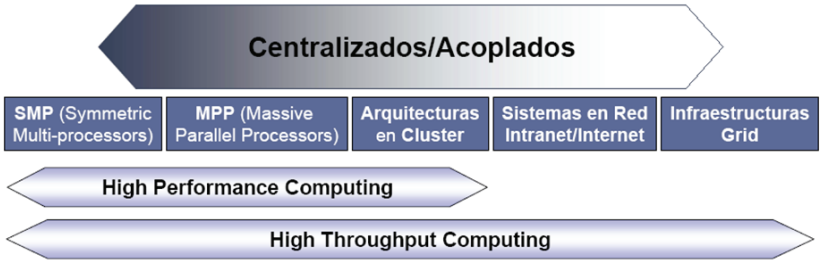

Figura 4.4. Entornos de computación paralela

La aplicación de los diferentes paradigmas de paralelismo sobre las metaheurísticas da lugar a lo que se llama modelos para el diseño de metaheurísticas paralelas. A este respecto, se han propuesto una gran cantidad de modelos paralelos tanto para las metaheurísticas basadas en trayectoria como para las basadas en población, de acuerdo con sus características intrínsecas [47-55]. Se puede distinguir entre tres niveles [123]: diseño de metaheurísticas paralelas a nivel de algoritmo, donde las metaheurísticas son gestionadas de manera independiente y cooperativa, de forma que con el diseño paralelo se busca aumentar la efectividad de las técnicas; a nivel de iteración, donde se busca acelerar la ejecución del algoritmo pero sin alterar su comportamiento; y a nivel de solución, donde se busca acelerar la parte más costosa del algoritmo, que normalmente tiene que ver con la evaluación de la calidad de una solución. La Tabla 4.1 resume esta clasificación.

Tabla 4.1. Clasificación de modelos paralelos para el diseño de metaheurísticas

Modelo paralelo	Dependencia del problema	Comportamiento	Granularidad	Objetivo
Nivel de algoritmo	Independiente	Se modifica	Metaheurística	Efectividad
Nivel de iteración	Independiente	Sin cambios	Iteración	Eficiencia
Nivel de solución	Dependiente	Sin cambios	Solución	Eficiencia

La citada taxonomía es complementaria a la de las estrategias paralelas, pero desde el punto de vista de las metaheurísticas en lugar de desde el punto de vista de las estrategias paralelas. Volviendo a éstas, la ejecución de algoritmos paralelos se puede enmarcar en dos grandes categorías. Por un lado está lo que se denomina computación de alto rendimiento, o HPC (*High Performance Computing*), cuyo objetivo es reducir el tiempo de ejecución de un único sistema paralelo. Existen varios modelos con los que desarrollar este tipo de aplicaciones (Figura 4.4), que van desde la memoria compartida (que utilizan por ejemplo sistemas SMP –*Symmetric Multi-Processing*) a la memoria distribuida (propia de las plataformas de computación distribuidas, como las infraestructuras grid), tanto en alternativas centralizadas (clusters con nodos maestro-esclavos) como descentralizadas (clusters independientes), si bien se puede decir que de entre todas las plataformas de computación paralela para ejecutar programas HPC destaca la computación clúster.

El otro gran paradigma de computación paralela lo forman las aplicaciones de alta productividad, o HTC (*High Throughput Computing*), cuyo objetivo es conseguir un alto número de ejecuciones independientes por unidad de tiempo. En este caso, el paradigma más representativo es la computación grid. La Figura 4.4 muestra los entornos de computación paralela para las aplicaciones HPC y HTC. Como se puede observar, la computación de alta productividad puede ser aplicada en todos los entornos, si bien tiene más sentido en los sistemas acoplados (distribuidos), ya que el paradigma está diseñado para aprovechar un gran número de recursos, y éstos suelen ser a priori mucho mayores en los sistemas acoplados que en los centralizados.

Sin embargo, un paso previo a la selección y diseño de la aplicación paralela que se implementará está el de hacer un estudio con el que se compruebe cuánto se ganará con la paralelización. Para esto hay que realizar un análisis previo donde se examine la complejidad numérica del código que se quiere paralelizar para determinar su escalabilidad, así como determinar el grado de paralelismo con el que se averiguará el número de procesos paralelos en los que se puede dividir el código y calcular la ganancia que se obtiene mediante la ley de Amdahl [179]. También se debe comprobar la granularidad de la implementación paralela para determinar el grado de acoplamiento requerido en la arquitectura subyacente, así como un análisis general del código que nos

permita determinar cómo, dónde y por qué paralelizar. Para más información sobre las medidas de rendimiento y análisis en computación paralela, consúltense las referencias [179, 180].

A continuación se expondrán brevemente los principios teóricos en los que se sustentan los dos modelos de paralelismo empleados en esta tesis, esto es, se introducirán los conceptos fundamentales sobre computación en clúster (HPC) utilizando MPI (*Message Passing Interface* [140]), y se explicará brevemente cómo se utiliza una grid real para computación HTC.

Computación en clúster con MPI

Un clúster es un conjunto de ordenadores homogéneos unidos con una red de alta velocidad para formar un sistema de computación de alto rendimiento (al trabajar todos los ordenadores en paralelo). Es un sistema seguro y fiable, ya que la red de alta velocidad está dedicada y controlada, las máquinas son totalmente homogéneas y, al tratarse de un sistema centralizado (hablamos por tanto de *clusters dedicados*), su administración es relativamente sencilla. Sin embargo, al ser una plataforma basada en memoria distribuida, a la hora de construir aplicaciones HPC, requiere modelos de programación de memoria distribuida, como son las técnicas de paso de mensajes entre las máquinas, y más concretamente MPI (*Message Passing Interface*).

MPI puede ser considerada hoy día la biblioteca estándar para la computación paralela basada en paso de mensajes. Es una biblioteca de libre distribución basada en un estándar de interfaces de programación de aplicaciones. Dicha biblioteca está formada por un amplio conjunto de primitivas que permiten el paso de mensajes para soportar de manera eficiente el procesamiento paralelo en un elevado número de procesadores conectados en una misma red de comunicaciones. Puede encontrarse amplia documentación sobre MPI en la web *MPI-Forum* [140]. De manera resumida, se puede decir que el estándar MPI incluye: comunicaciones punto a punto entre dos equipos, operaciones colectivas entre varios equipos, agrupamiento de procesos, topologías de comunicación y soporte para lenguajes de programación como C y

Fortran. Por el contrario, no incluye: soporte para el control de tareas, soporte para hilos, ejecución remota de procesos, soporte de sistemas operativos para la recepción de interrupciones y comunicaciones de memoria compartida en un mismo equipo. Por tanto MPI es un estándar ampliamente difundido que incluye un alto número de funciones probadas exhaustivamente y que dispone de un gran número de implementaciones.

En el modelo de programación MPI, un cómputo o tarea está formado por uno o más procesos, comunicados a través de llamadas a rutinas de librerías para mandar (*send*) o recibir (*recv*) mensajes de otros procesos. Por las propias características de MPI, uno de los modelos más habituales es el modelo *Maestro-Esclavo*, siendo el proceso maestro, que normalmente tiene identificador 0, el que se encarga de realizar la organización del trabajo o la recopilación de los datos que el resto de procesos (procesos esclavos) deben llevar a cabo. En una aplicación sencilla, el proceso maestro procedería a enviarle los datos a los procesos esclavos, éstos operarían con ellos y después le enviarían de nuevo los resultados al proceso maestro, que interpretará los datos de manera adecuada. La Figura 4.5 muestra el esquema básico de ejecución que se ha descrito con un ejemplo de 4 procesadores.

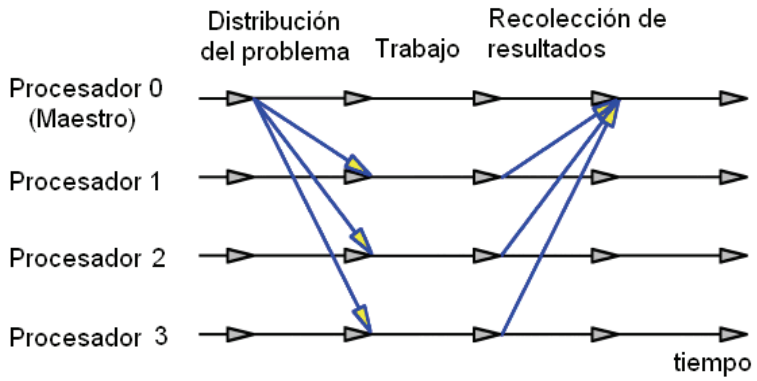

Figura 4.5. Esquema básico de distribución de tareas en un modelo maestro-esclavo con cuatro procesadores

Todo programa escrito con esta librería tiene un esqueleto similar, aunque la estructura varía dependiendo del lenguaje de programación base que se utilice (en el caso que se describe en esta tesis, C). Tal y como se puede ver en el esquema de la Figura 4.6, hay 4 fases bien diferenciadas: Fase 1: Incluir la

librería de funciones que contiene la biblioteca de MPI (con unas 115 funciones). Fase 2: Inicializar MPI. Antes de poder usar cualquier comando de MPI se debe haber realizado la inicialización correspondiente usando el comando *MPI_Init*. Fase 3: Esta fase se corresponde con el cuerpo del programa, donde se desarrolla la implementación del mismo de modo similar a cualquier programa C, pudiendo hacer uso de los comandos específicos de MPI. Las funciones *MPI_Comm_size* (indica el número de procesos disponibles para la ejecución del programa) y *MPI_Comm_rank* (indica el identificador del proceso con el que se trabaja) forman parte del cuerpo, pero su uso es obligatorio, por lo que las hemos incluido en el esquema para resaltarlas. Por último, fase 4: Finalización de MPI. Una vez que se ha implementado el programa y ya no se usan más los comandos MPI, se debe invocar *MPI_Finalice.*

```
# include <mpi.h>                          // FASE 1

main (int argc, char **argv) {

        int nproc;     /* Número de procesos */
        int mi_dir;    /* Mi dirección: 0<=mi_dir<=(nproc-1) */

        MPI_Init (&argc, &argv);            // FASE 2

        // FASE 3:
        MPI_Comm_size (MPI_COMM_WORLD, &nproc);
        MPI_Comm_rank(MPI_COMM_WORLD, &mi_dir);

        /* CUERPO DEL PROGRAMA */

        MPI_Finalize();                     // FASE 4
}
```

Figura 4.6. Estructura básica de un programa con MPI

Todos los flujos creados con MPI se ejecutan de forma independiente en los procesadores a los que son enviados. Queda como labor del programador el dividir de forma adecuada los segmentos de código que se quieren paralelizar para que se ejecuten de manera simultánea en los distintos procesadores del clúster. Una buena idea de optimización de rendimiento es la de combinar MPI con openMP [181], que es una librería de programación paralela para memoria compartida que permite paralelizar el código de un programa entre los distintos cores de un nodo del clúster, de forma que podría utilizarse MPI para enviar distintos mensajes a los diferentes nodos de un clúster y dentro de cada nodo

utilizar openMP para realizar un paralelismo aún más fino que utilizara de la forma altamente eficiente cada uno de los cores de ese nodo.

Las dos instrucciones más representativas de comunicación en MPI son: *MPI_Send* (para enviar datos) y *MPI_Recv* (para recibirlos). Junto a las cuatro instrucciones que aparecen en la Figura 4.6, éstas pueden considerarse las instrucciones más relevantes de la librería. Además, es importante remarcar que las instrucciones de MPI (y en general de cualquier técnica de paso de mensajes) se pueden dividir en aquellas que provocan dos tipos de comunicaciones:

- Comunicaciones bloqueantes: Son comunicaciones donde el proceso que realiza el envío, o que espera algún mensaje, permanece bloqueado hasta que la comunicación ha finalizado.
- Comunicaciones no bloqueantes: Son comunicaciones donde el proceso que realiza el envío o que espera algún mensaje continúa sin esperar a que la comunicación haya finalizado, de manera que éste no se bloquee.

Esta clasificación es independiente de otras clasificaciones, como por ejemplo la que divide las comunicaciones entre las que son punto a punto y las que son colectivas. En cualquier caso, existe documentación muy extensa de las diferentes funciones y detalles que forman parte de MPI (especificación del mensaje, formatos de las funciones, parámetros, excepciones…). Para obtener más información, consúltese la referencia [140], donde se encontrará información detallada sobre todos los aspectos de la biblioteca.

Computación HTC utilizando grid

La grid es un servicio para compartir potencia de cálculo y capacidad de almacenamiento a través de la red. El punto clave es la abstracción y la virtualización de los recursos que componen la infraestructura. De este modo, para el usuario es como interactuar con un único y potente ordenador. La Grid toma el nombre de su analogía con la red eléctrica (en inglés *power grid*) donde los conceptos de transparencia para el usuario, infraestructura global y utilidad

son claves. El origen de los sistemas grid se puede establecer a principios de los años 90, originalmente como un proyecto para conectar grandes supercomputadores geográficamente distribuidos en EE.UU. Sin embargo, el concepto "grid" aún no existía y se utilizaban entonces términos como "metasistema" (*metasystem*) o "metacomputador" (*metacomputing*) para referirse a los recursos computacionales disponibles de forma transparente al usuario mediante redes" [182].

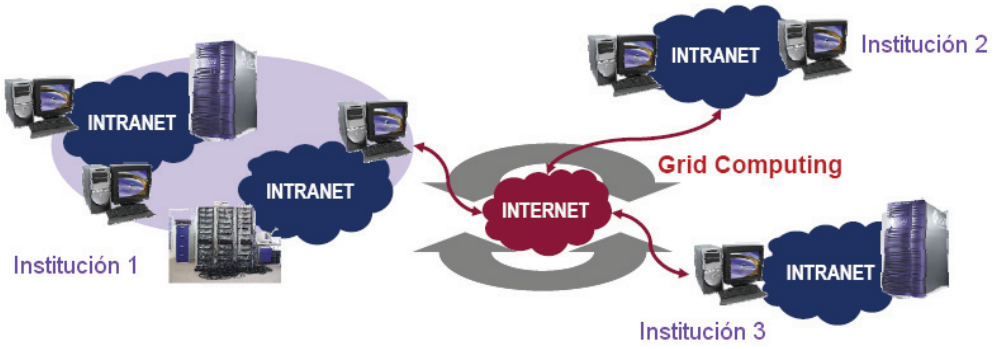

Figura 4.7. Esquema de una posible grid formada por 3 instituciones

Aunque aún en la actualidad no existe una definición firme y formal de lo que es la grid, sí que hay una lista de características comunes que está ampliamente aceptada por la comunidad científica. De acuerdo con Ian Foster, que puede ser considerado el padre de la grid, esta tecnología es un sistema que [183]:

(1) Coordina recursos que no están sujetos a un control centralizado.

(2) Utiliza protocolos e interfaces estándar, abiertos y de propósito general.

(3) Proporciona diferentes calidades de servicio no triviales.

Así pues, la tecnología grid interconecta recursos en diferentes dominios de administración respetando sus políticas internas de seguridad y su software de gestión de recursos en la intranet (ver Figura 4.7). Es una tecnología dentro del área global de computación de altas prestaciones que satisface las demandas de aplicaciones que requieren alta productividad (HTC) y que tienen enormes necesidades de cómputo y/o almacenamiento.

Otra definición más estructurada expuesta por los precursores de la computación grid en [184], plantea la existencia de organizaciones virtuales (*virtual organizations* –VO) como puntos de partida de este enfoque. Una VO es "un conjunto de individuos y/o instituciones definida por reglas que controlan el modo en que comparten sus recursos". Básicamente, son organizaciones unidas para lograr objetivos comunes. Ejemplos de VOs podrían ser proveedores de servicios de aplicaciones o almacenamiento, equipos de trabajo empresarial realizando análisis estadístico de datos y planeamiento estratégico o universidades involucradas en un proyecto de investigación conjunto.

Sin embargo, por ser un sistema de cómputo distribuido y heterogéneo, la computación grid presenta problemas que son típicos de éstos. Por ejemplo, al tratarse de un sistema dinámico, los recursos cambian, y por lo tanto se deben utilizar sistemas que se encarguen de buscar recursos disponibles (meta-planificadores), ya que de la misma forma que se pueden agregar nodos que aporten recursos, más tarde pueden desaparecer, haciendo que se reduzcan los reduciéndose los recursos que aportaban y el software que pudieran tener. Existen varios entornos de acceso a la grid, si bien la tecnología más utilizada es Globus [185], que incluye el meta-planificador GridWay [186], si bien hay otras opciones, como gLite [187] (que incluye el meta-planificador WMS [188]), Condor [189] o Proactive [190].

A continuación hablaremos de manera muy resumida sobre el modelo de programación que se sigue en computación Grid, es decir, hablaremos acerca de cómo son las aplicaciones que se ejecutan en este tipo de infraestructuras. Como se ha dicho al principio del capítulo, la computación grid está pensada especialmente para aplicaciones de alta productividad (HTC). De acuerdo con el tipo de ejecución que siguen los trabajos que se ejecutan en la grid, se pueden distinguir tres modelos de aplicaciones:

- El modelo HTC síncrono, donde todos los trabajos (denominados *jobs* en el ámbito de la computación grid) se envían de forma simultánea a la grid, recibiendo el resultado de cada cuando terminan.

- El modelo maestro esclavo (*master-slave*), donde hay un trabajo maestro que pre procesa los resultados de los demás trabajos (esclavos) antes de obtenerse el resultado final.

- Los flujos de trabajo, donde las entradas y las salidas de los trabajos forman un flujo de ejecución, de manera que para que comience un trabajo deben haber finalizado otros (esto a veces genera esperas).

Una ilustración de estos tres modelos puede observarse en la Figura 4.8.

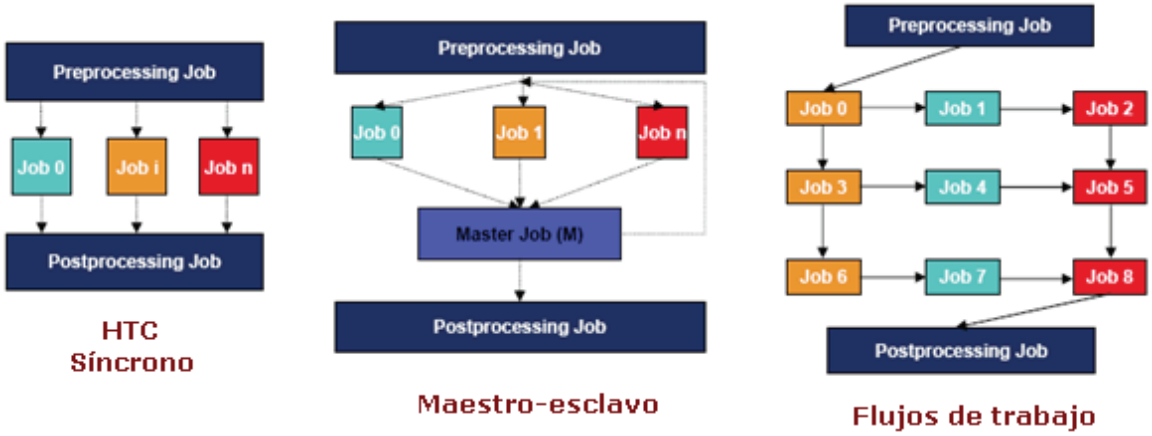

Figura 4.8. Modelos de ejecución de trabajos en los sistemas HTC

En conclusión, la tecnología grid está especializada en la ejecución eficiente de aplicaciones que requieren un alto nivel de productividad. A estas aplicaciones se las denomina aplicaciones grid, y para aprovechar las características de la grid, deben requerir mucha capacidad de cómputo o/y de almacenamiento. Sin embargo, los recursos de la grid son heterogéneos y pueden desaparecer o incorporarse al sistema sin previo aviso, por lo que se hace necesario un software que controle la infraestructura (*middlewares* y planificadores) y la implementación de modelos y topologías de paralelismo complejos, donde el paso de mensajes es muy frecuente no resulta eficiente. Los modelos más apropiados para paralelizar las aplicaciones son los pensados para aplicaciones de alta productividad (Figura 4.8). En concreto, en esta tesis se ha utilizado un modelo maestro-esclavo para ejecutar varias instancias de las metaheurísticas diseñadas a la vez. Este modelo paralelo ofrece varias ventajas, ya que se trata de un modelo conceptualmente simple que minimiza el número de comunicaciones y garantiza el control de las ejecuciones que se lanzan en paralelo: el maestro envía las tareas a los esclavos, que ejecutan las distintas metaheurísticas y devuelven el resultado al maestro cuando acaban, el cual las examina y las procesa adecuadamente.

4.5. Resumen

En este capítulo de la tesis se han descrito de manera general las metodologías que se han aplicado para abordar el problema de optimización seleccionado. Se ha comenzado enmarcando las estrategias de optimización utilizadas, las metaheurísticas, y justificando su utilización debido a la complejidad del problema abordado. A continuación se han definido de manera general dichas técnicas y se ha realizado una clasificación de las mismas atendiendo a la forma que tienen de explorar el espacio de búsqueda. Después se han descrito sus principales características, haciendo especial hincapié en las metaheurísticas desarrolladas para esta tesis. Por otra parte, debido al carácter estocástico de dichos métodos, se ha hecho necesario realizar un completo análisis estadístico para proporcionar validez científica a los datos obtenidos a partir de las citadas metaheurísticas. La metodología utilizada en dicho análisis también se ha explicado en este capítulo.

Además, como se explicó en el capítulo 3 de este documento, se ha abordado un problema real de ingeniería utilizando instancias de grandes dimensiones, por lo que resulta muy conveniente hacer uso de técnicas de paralelismo con las que mejorar el rendimiento de las metaheurísticas (a varios niveles). Por tanto, en este capítulo también se han dado explicaciones acerca de estrategias de paralelismo aplicadas a las metaheurísticas, tanto diseñadas para aumentar su rendimiento (MPI) como para mejorar la productividad (grid).

Tras la explicación de los fundamentos teóricos de las técnicas que se han utilizado para resolver el problema de optimización abordado en esta tesis, en el siguiente capítulo se explicarán de manera concreta los detalles y el diseño específico final que se implementó para que los algoritmos desarrollados consiguieran soluciones de alta calidad en la resolución del problema real que se ha abordado en nuestra investigación.

Capítulo 5

Diseño y desarrollo de algoritmos

En este capítulo se explica el diseño detallado y los ajustes realizados a cada uno de los algoritmos desarrollados para adaptar sus modelos genéricos (introducidos en el capítulo anterior) al problema de optimización abordado en esta tesis. A este respecto, se ha de indicar que los primeros pasos de la investigación se dieron utilizando instancias de prueba del problema FAP, lo cual condujo a que las primeras versiones de los algoritmos, tanto en sus versiones secuenciales como paralelas, tuvieran un esquema más sencillo que los que finalmente se presentan en este capítulo, sin embargo, se ha considerado que dichos diseños intermedios no presentan tanta relevancia como las versiones finales de los mismos. Por tanto, los esquemas de los algoritmos que se describirán a continuación se corresponden con las últimas actualizaciones de los mismos. Dichos diseños han hecho posible la obtención de resultados de muy alta calidad en la resolución de instancias reales del problema que se describió en el capítulo 3 de esta tesis. Además, para mejorar el rendimiento de las metaheurísticas desarrolladas, todas ellas han sido combinadas con un eficiente método de búsqueda local. Este método comenzó también siendo un sencillo algoritmo de mejora de las soluciones, y pasó por una serie de mejoras hasta que finalmente se adoptó un eficiente método de búsqueda intensiva que se ajusta de manera altamente eficiente al problema que se aborda en esta tesis.

El presente capítulo ha sido dividido en tres grandes partes. En la primera se describirán un conjunto de generalidades que han sido aplicadas a todos los algoritmos, como son la descripción del individuo utilizado, la explicación de la búsqueda local con la que se combinan todas las metaheurísticas o los operadores genéticos (cruce y mutación) que son utilizados por aquellos

métodos que los incluyen. Tras las generalidades, se describirán las metaheurísticas que finalmente han sido incluidas en nuestro estudio. A este respecto, se ha tratado de seleccionar un conjunto heterogéneo de metaheurísticas que abordaran la resolución del problema propuesto en esta tesis desde varios enfoques, de manera que la exploración del espacio de búsqueda fuera lo más diversa y exhaustiva posible. Se han desarrollado técnicas clásicas, pero también novedosas, técnicas basadas en trayectoria y también en población, así como técnicas basadas en diferentes modelos (probabilístico, colonias de insectos, evolutivo…). Sin embargo, en este capítulo no se explicará el esquema general de cada estrategia, ya que éstos se encuentran ampliamente detallados en la bibliografía. Lo que se describirá en las siguientes secciones serán las características específicas y los detalles de las versiones ajustadas al problema real que se resuelve en esta tesis. Finalmente, el capítulo concluirá con la explicación de un eficiente modelo paralelo que se diseñó y desarrolló para mejorar los resultados obtenidos por las versiones secuenciales de los algoritmos. A este respecto, también hay que añadir que el sistema paralelo propuesto en este capítulo ha sido fruto de una larga evolución que nos llevó a hacer pruebas con diversos modelos paralelos que fueron aplicaron a distintas metaheurísticas desarrolladas. Al igual que en el caso de las metaheurísticas, la hiperheurística que se explica en este capítulo representa la conclusión de un extenso trabajo realizado con diversas técnicas paralelas aplicadas al dominio de las técnicas metaheurísticas.

5.1. Consideraciones comunes

En este apartado se explicarán las consideraciones que son comunes a todos los algoritmos desarrollados con el fin de no repetirlos en cada sección del capítulo. Como se ha indicado anteriormente, todas las metaheurísticas utilizan la misma codificación para las soluciones (todos los individuos de todos los algoritmos tienen la misma representación). Por otro lado, el algoritmo de búsqueda local con el que se combinan todas las metaheurísticas para mejorar las soluciones en cada iteración del algoritmo también es una parte común que se explicará en

esta sección. Finalmente, hay ciertos operadores genéticos que se repiten en varias metaheurísticas y que por tanto se explican únicamente en esta sección.

Codificación de las soluciones

Según la formulación matemática del problema explicada en la sección 3.3, una solución al problema real FAP que se maneja en esta tesis se obtiene asignando a cada uno de los TRX de la red, $t_i \in T$ una frecuencia válida de F_i (ver capítulo 3.3 de esta tesis: *Formulación matemática de un caso real*). Por tanto, los individuos de todos los algoritmos desarrollados se codificarán con un *array* de enteros, p, donde $p(t_i) \in F$ es la frecuencia que se asigna al TRX t_i de la red. Por tanto, la solución al problema será la planificación completa para cada uno de los TRX de la red, especificando qué frecuencia debe utilizar cada uno para minimizar las interferencias que se producen. La Figura 5.1 muestra gráficamente el individuo que usarán todos los algoritmos.

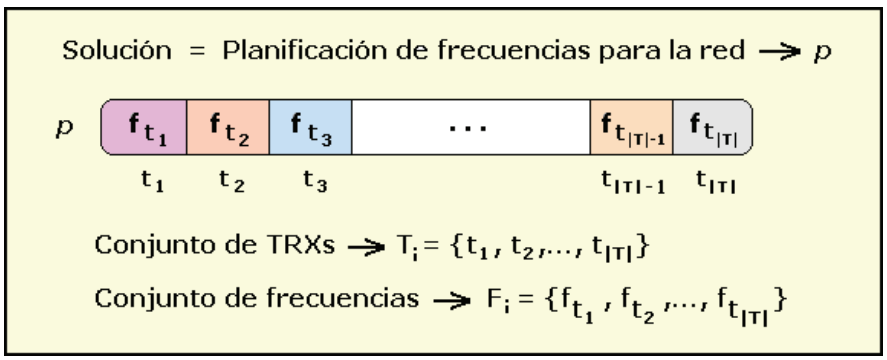

Figura 5.1. Codificación del individuo para todos los algoritmos

A lo largo de la investigación realizada, el individuo se ha ido mejorando, de forma que se ha ido adaptando para que sea lo más estándar posible, y pueda ser utilizado de manera sencilla por un gran conjunto de algoritmos. Otras representaciones válidas que se han utilizado en nuestra investigación consistía en representar a las soluciones como matrices de booleanos, en lugar de cómo vectores de enteros, de forma que la fila tuviera la misma función que los índices del vector y la columna representara las frecuencias que podían ser

seleccionadas. Sin embargo, esta solución se descartó porque para problemas reales, es menos eficiente que la adoptada mediante un vector de enteros.

Búsqueda local

Los métodos de búsqueda local (*Local Search* –LS, [28, 41]) son estrategias heurísticas que realizan una búsqueda más o menos intensiva en el vecindario de una solución con el objetivo de mejorar la calidad de ésta en un periodo relativamente corto de tiempo. Estas técnicas suelen estar hibridizadas con otras heurísticas más complejas actuando como métodos de mejora complementarios a los operadores propios de las estrategias con las que se combinan. Además, debido al carácter estocástico de las metaheurísticas, las estrategias de LS son un elemento prácticamente obligatorio cuando se manejan problemas de ciertas dimensiones y quieren obtenerse soluciones competitivas. El método de búsqueda local que se explica en esta sección (Algoritmo 5.1) ha sido especialmente diseñada para ser aplicada al problema FAP que se maneja en esta tesis [120] y es un elemento común para todos los algoritmos que se describen en este capítulo. En este punto, debemos señalar que en el marco de la investigación realizada se diseñaron otros procedimientos de búsqueda local que funcionaban peor que el que se incluye en este capítulo, y que por tanto no han sido incluidos en este documento. La búsqueda local que aquí se describe fue fruto de la colaboración mantenida dentro del proyecto OPLINK con otros investigadores del consorcio, y más concretamente con el nodo de la Universidad de La Laguna [191].

El funcionamiento de la LS descrita en el Algoritmo 5.1 se basa en la optimización en la asignación de frecuencias de un sector concreto sin cambiar las asignaciones en el resto de la red. Existen otros diseños de búsqueda local más sencillos donde únicamente se intercambia la asignación en uno de los TRXs de la planificación, pero el diseño utilizado en esta tesis resulta más exhaustivo y las mejoras que se producen son más significativas. En la solución codificada, los vecinos de una solución candidata se obtienen cambiando las frecuencias de los TRXs que se encuentran en un sector siguiendo el siguiente método (Algoritmo

5.1): en primer lugar, se ordenan las frecuencias disponibles para el sector que se está manejando de acuerdo con su coste asociado. En el siguiente paso se considera tanto la asignación de una frecuencia permitida que genera un menor coste para un TRX dentro del sector con el que se trabaja, como las asignaciones de las frecuencias adyacentes a ésta (siempre que se permita tal asignación según los requisitos del problema). Este proceso se repite para todos los TRXs del sector que se está explorando. Dentro de todas las soluciones generadas, aquella que presenta un coste global menor, es considerada como una nueva solución vecina a la actual, mientras que el resto de soluciones generadas en este proceso se descartan. Además, no es necesario volver a calcular el *fitness* de la solución completa, ya que éste se puede hallar teniendo en cuenta los cambios realizados en el sector manipulado, con el consiguiente ahorro de tiempo.

Algoritmo 5.1 – Pseudocódigo de la búsqueda local

```
1:  Entrada: Una solución, p
2:  sigSectores ← {1, …, numSectores}
3:  mientras (sigSectores != Ø) hacer
4:      actualSectores ← sigSectores
5:      sigSectores ← Ø
6:      mientras (actualSectores != Ø) hacer
7:          sector ← seleccionar_sector_aleatorio (actualSectores)
8:          vecino ← reasignar_frecuencias_de_p (sector)
9:          si (vecino es_mejor_que p) entonces
10:             p ← vecino
11:             sigSectores += añadir_sectores_interferidos_por (sector)
12:             sigSectores += añadir_sectores_que_interfieren_a (sector)
13:         fin si
14:     fin mientras
15: fin mientras
16: devolver p
17: Salida: Una solución p mejorada
```

Los vecinos de la solución que se está manejando son determinados aleatoriamente (línea 7), sin embargo, se trata de evitar cualquier vecino que no mejore a la solución actual (líneas 9 y 10). Cuando el algoritmo comienza, se consideran todos los sectores de la solución procesada (líneas 2 y 4), seleccionándose de manera aleatoria en cada iteración del algoritmo un nuevo

sector cuyas frecuencias serán reasignadas (líneas 7 y 8). En cada movimiento, si se obtiene un vecino mejor que la solución actual, *p*, ésta es sustituida por ese vecino (línea 10), añadiendo a la lista de sectores que deben ser examinados los que interfieren o son interferidos por el sector que causó la mejora (líneas 11 y 12). Cuando los sectores que se están analizando acaban (línea 6), los sectores que se han añadido fruto de la mejora de soluciones pasan a ser analizados (línea 4) y se vacía el conjunto de sectores a analizar en la siguiente iteración (línea 5). El método de LS terminará cuando ninguno de los vecinos mejore a la solución actual (línea 3). Al final del proceso, el método de búsqueda local devolverá una solución, *p*, mejorada respecto a la solución que tuvo como entrada (línea 16).

Operadores genéticos comunes

En esta sección se explican los operadores genéticos que se utilizan de manera común en las metaheurísticas que requieren su uso. Así, todos los algoritmos realizan una inicialización aleatoria de sus soluciones (ya sea de una población entera o de un único individuo), por lo que para realizar esta operación se realizará un recorrido por todos los TRX del individuo (Figura 5.1) asignando a cada uno de ellos una frecuencia válida que será obtenida de manera aleatoria.

Algoritmo 5.2 – Pseudocódigo de la mutación aleatoria

```
1: Entrada: Una solución, p y una probabilidad de mutación Prob_mut
2: para (t_i ∈ p) hacer
3:     alea ← rand [0, 1]
4:     si (alea < Prob_mut) entonces
5:         f ← rand (F_i)
6:         p(t_i) ← f
7:     fin si
8: fin para
```

Otra de las operaciones comunes a la mayoría de los algoritmos desarrollados es la mutación, ya que ésta es una operación fundamental para las

metaheurísticas basadas en trayectoria, y su uso controlado mejora habitualmente las soluciones que se obtienen en las metaheurísticas basadas en población. Se han hecho pruebas con múltiples algoritmos de mutación, cambiando el criterio de alteración de las soluciones en mayor o menor medida, sin embargo, la experiencia ha demostrado que el método de mutación que mejores resultados obtiene es la mutación aleatoria, por tanto éste ha sido el método de mutación utilizado por las metaheurísticas que lo han requerido. El único parámetro para esta función es una probabilidad de mutación ($Prob_{mut}$, Algoritmo 5.2), que debe ser superada para que se realice la operación. También se ha demostrado de forma empírica que el nivel al cual funciona mejor esta operación es a nivel de TRX (más que a nivel de sector o aplicando este operador a otros fragmentos mayores), por lo que se ha aplicado en todos los casos una mutación aleatoria sin modificaciones, tal y como se describe en el pseudocódigo del Algoritmo 5.2.

Por otro lado, tanto el algoritmo genético como la búsqueda dispersa utilizan un método de combinación de soluciones, un cruce de individuos, que en ambos casos es el cruce uniforme (*uniform crossover*, UX) a nivel de TRX, tal y como se describe en el Algoritmo 5.3.

Algoritmo 5.3 – Pseudocódigo del cruce uniforme

```
1:  Entrada: Dos soluciones padre, p1 y p2
2:  para (t_i ∈ d) hacer
3:      alea ← rand [0, 1]
4:      si (alea < 0,5) entonces
5:          d(t_i) ← p1(t_i)
6:      sino
7:          d(t_i) ← p2(t_i)
8:      fin si
9:  fin para
10: Salida: Una solución hijo, d
```

Por último, la actualización de soluciones llevada a cabo por todos los algoritmos (en caso de los de trayectoria sólo se actualiza una única solución) siempre se realiza aplicando una técnica elitista, donde las mejores soluciones siempre pasan a la siguiente generación, reemplazándose los individuos de peor

calidad (los que tienen un coste más alto) en cada iteración del algoritmo. En caso de las metaheurísticas basadas en trayectoria se reemplaza la solución actual sólo si la nueva tiene mejor calidad que la actual (un coste menor).

5.2. Metaheurísticas desarrolladas

En esta sección se explican los detalles de diseño de cada una de las técnicas metaheurísticas que se incluyen en esta tesis. Cada metaheurística ha sido cuidadosamente ajustada para obtener resultados de calidad para el problema FAP descrito en el capítulo 3 de *Fundamentos del problema*, por lo que en los siguientes apartados no se explicarán los esquemas genéricos de los algoritmos (que se pueden encontrar en la bibliografía), sino las versiones adaptadas que fueron finalmente desarrolladas para resolver el problema abordado en nuestra investigación. Como ya se ha mencionado en capítulos anteriores, se han desarrollado cuatro metaheurísticas basadas en población (PBIL, SS, GA, y ABC) y otras tres basadas en trayectoria (ILS, VNS y GRASP). Todas ellas hacen uso tanto del mismo individuo como de las estrategias comunes (iniciación de individuos, búsqueda local, mutación, etc.) explicadas en la sección anterior.

Aprendizaje incremental basado en población

El aprendizaje incremental basado en población (PBIL –*Population Based Incremental Learning*) [36, 37] es una metaheurística que combina estrategias evolutivas con un modelo probabilístico basado en aprendizaje competitivo para la resolución de problemas de optimización. De esta manera, el funcionamiento del algoritmo se basa en la evolución de una distribución de probabilidad, *P*, que es representada por un vector de probabilidad con tantas posiciones como TRX tengan las planificaciones de frecuencias de las soluciones (codificadas como se explica en la Figura 5.1). En cada posición de *P* se almacenan las probabilidades que tiene cada frecuencia válida de ser asignada al TRX que es identificado por

esa posición del vector (por tanto, el vector *P* se ha codificado realmente como una matriz). El Algoritmo 5.4 resume el funcionamiento del algoritmo.

Algoritmo 5.4 – Pseudocódigo de PBIL

```
1:  P ← inicializar_vectorProb
2:  población ← generar_población (P)
3:  mientras (! límite_tiempo) hacer
4:      población ← búsqueda_local (población)
5:      mejorIndiv ← seleccionar_mejor_individuo (población)
6:      P ← actualizar_vectorProb (P, mejorIndiv)
7:      P ← mutar_vectorProb (P)
8:      población ← regenerar_población (P, población)
9:  fin mientras
10: devolver seleccionar_mejor_individuo (población)
```

A diferencia del resto de metaheurísticas, PBIL genera la población a partir de su vector de probabilidad, *P*, que debe ser inicializado al comienzo del proceso (línea 1) con la misma probabilidad para todas las frecuencias, ya que éstas deben tener al principio las mismas oportunidades de ser seleccionadas a la hora de crear las primeras soluciones. Tras este proceso de inicialización, la primera población es generada (línea 2) haciendo uso del vector *P*. A continuación, se aplica el método de búsqueda local explicado en Algoritmo 5.1 a cada planificación de frecuencias de la población (línea 4, Algoritmo 5.4). El siguiente paso consiste en seleccionar al mejor individuo de la población (línea 5) para actualizar con él el vector de probabilidad (línea 6), para hacer que éste evolucione con la influencia de la solución de mayor calidad (con un coste más bajo) generada hasta la fecha. Esta actualización se lleva a cabo siguiendo la siguiente expresión: $P_i \leftarrow P_i \times (1{,}0 - LR) + mejorIndiv_i \times LR$. Como se puede ver, *P* modifica su valor de forma que la probabilidad de seleccionar la frecuencia que coincide con la del mejor individuo sube, dependiendo del valor del parámetro *LR* (*learning rate*), que es el factor de aprendizaje del algoritmo, mientras que el resto de frecuencias disminuyen su probabilidad equitativamente. Después, para alterar la evolución del vector de probabilidad, se realiza una mutación del mismo (línea 7). Esta operación dependerá de los parámetros *MutP* (*mutation probability* –probabilidad de mutación) y *MutA* (*mutation amount* –cantidad de

mutación), y se realizará atendiendo al pseudocódigo descrito en el Algoritmo 5.5.

Algoritmo 5.5 – Pseudocódigo de la mutación del vector de probabilidad

```
1:  Entrada: El vector de probabilidad, P, y los parámetros MutA y MutP
2:  para (las probabilidades de las frecuencias de cada TRX: P_i ∈ P) hacer
3:      alea ← rand [0, 1]
4:      si (alea < MutP) entonces
5:          P_i_previo ← P_i
6:          P_i ← P_i × (1,0 – MutA) + rand [0, 1] × MutA
7:          para (las probabilidades del resto de frecuencias de P_i) hacer
8:              Restar equitativamente la probabilidad ganada por P_i
9:          fin para
10:     fin si
11: fin para
```

Finalmente, con los nuevos valores del vector de probabilidad se generará una nueva población (línea 8, Algoritmo 5.4), teniendo en cuenta que la mejor solución de la población actual se conservará en la nueva población generada (se ha aplicado elitismo en la regeneración de la población). Al final del proceso, cuando el límite temporal del algoritmo llega a su fin (línea 3, Algoritmo 5.4), la metaheurística devuelve la mejor solución de la población como solución final (línea 10, Algoritmo 5.4).

Búsqueda dispersa

La búsqueda dispersa (SS –*Scatter Search*) [32-35] es una metaheurística basada en población que se ha demostrado como una de las técnicas más eficientes en la resolución del problema FAP. El algoritmo trabaja con un conjunto reducido y representativo de soluciones tomadas de la población, que es llamado conjunto de referencia (*refSet –reference set*). Las soluciones se codifican tal y como se ha explicado anteriormente (Figura 5.1), como *arrays* de enteros que representan una planificación de frecuencias para el problema

descrito. En el Algoritmo 5.6 se describe el funcionamiento general de la metaheurística.

Algoritmo 5.6 – Pseudocódigo de la búsqueda dispersa

```
1:  población ← inicializar_población_aleatoria
2:  población ← búsqueda_local (población)
3:  refSet ← generar_conjunto (población)
4:  mientras (! límite_tiempo) hacer
5:      subCjto ← generar_subCjtos (refSet)
6:      descendencia ← combinar_subCjtos (subCjto)
7:      descendencia ← búsqueda_local (descendencia)
8:      refSet ← actualizar_refSet (descendencia, población)
9:  fin mientras
10: devolver mejor_individuo (refSet)
```

El *refSet* contiene por un lado las mejores soluciones de la población (las planificaciones de frecuencia con un coste más bajo) y por otro las soluciones más diversas (las planificaciones de frecuencia que se distinguen más con las mejores). El algoritmo comienza con la generación aleatoria de la población, asignándole a cada TRX de la solución una frecuencia válida (línea 1). A continuación, se aplica el método de búsqueda local explicado en el Algoritmo 5.1 a cada uno de los individuos de la población (línea 2, Algoritmo 5.6). En el siguiente paso se genera el *refSet* (línea 3) seleccionando de una parte las mejores soluciones de la población y de otra las más diversas. Para calcular el grado de diversidad de dos soluciones respecto de una tercera, se calcula la distancia TRX a TRX de ambas soluciones respecto a esta tercera solución. La distancia se medirá contando dentro de cada sector de las dos soluciones que se comparan el número de TRX que difieren. Tras la generación del *refSet*, el siguiente paso consistirá en generar subconjuntos a partir de todos los elementos alojados en este conjunto (línea 5). Los subconjuntos se forman eligiendo todas las posibles parejas de soluciones del conjunto de referencia. A continuación se genera la descendencia a partir de todos los subconjuntos creados. La forma de crear nuevos individuos es combinar mediante cruce uniforme (Algoritmo 5.3) a los integrantes de cada subconjunto (línea 6, Algoritmo 5.6). Finalmente, se vuelve a aplicar la búsqueda local (Algoritmo 5.1) a todos los descendientes (línea 7, Algoritmo 5.6). El *refSet* es entonces

actualizado con las nuevas soluciones de manera que las mejores planificaciones de frecuencias permanezcan en él (línea 8). Tras realizar los experimentos de ajuste del algoritmo, se demostró que la mejor configuración consistía en almacenar únicamente la mejor planificación de frecuencias en el *refSet*, dejando que el resto de elementos del conjunto lo conformaran soluciones diversas. De esta manera, se mantenía la calidad en el conjunto de referencia, maximizando el espacio de búsqueda explorado. Por tanto, cuando en cada iteración del algoritmo se actualiza el conjunto de referencia, únicamente la mejor solución generada hasta ese momento se guarda en el nuevo *refSet*. El resto de soluciones del conjunto serán las |*refSet*|-1 planificaciones de frecuencia de la población que más difieran con la mejor (línea 8). El algoritmo repetirá este proceso hasta que el límite temporal del mismo sea alcanzado (línea 4), momento en el que se devolverá la mejor planificación de frecuencias encontrada como solución final (línea 10).

Algoritmo genético

A tenor de la cantidad de trabajos publicados en la bibliografía, se puede considerar a esta clásica metaheurística como el más conocido y utilizado de los algoritmos evolutivos (EA –*Evolutionary algorithms* [31]).

Algoritmo 5.7 – Pseudocódigo del algoritmo genético

```
1:  población ← inicializar_población_aleatoria
2:  población ← búsqueda_local (población)
3:  mientras (! límite_tiempo) hacer
4:      padres ← torneo_binario (población)
5:      descendencia ← cruce_uniforme (padres)
6:      descendencia ← mutación_aleatoria (descendencia)
7:      descendencia ← búsqueda_local (descendencia)
8:      población ← actualizar_población (descendencia)
9:  fin mientras
10: devolver mejor_individuo (población)
```

El funcionamiento del algoritmo genético (GA –*Genetic Algorithm*) [29, 30] está inspirado en la teoría de la evolución natural. Las soluciones manejadas por el GA son planificaciones de frecuencias que dan solución al problema abordado, tal y como se ha descrito anteriormente (Figura 5.1), están codificadas como *arrays* de números enteros. En el Algoritmo 5.7 se describe resumidamente el funcionamiento del mismo. Como se puede observar, el algoritmo comienza con una generación aleatoria de la población, de manera que se le asigna una frecuencia válida elegida aleatoriamente a cada uno de los TRXs de cada individuo de la población (línea 1). Tras este proceso, la búsqueda local descrita en el Algoritmo 5.1 se aplica a cada uno de los individuos aleatoriamente generados, de manera que todos sufren una mejora en su calidad, eliminando las interferencias con mayor coste (línea 2, Algoritmo 5.7).

Como se relatará en el siguiente capítulo, se han realizado numerosos experimentos para ajustar el algoritmo, entre los que se encuentran los que se hicieron para determinar el mejor método de selección de padres para generar la descendencia en cada generación. Se ha demostrado empíricamente que el mejor método de selección es el torneo binario, donde para cada padre se seleccionan de manera aleatoria dos candidatos de la población de los cuales se elije el mejor, es decir, el que tiene un coste más bajo en su planificación de frecuencias (línea 4). Tras este proceso, se obtendrá una descendencia de individuos que serán utilizados en la siguiente generación del algoritmo siempre y cuando mejoren a las peores soluciones de la población actual. El algoritmo utiliza el cruce uniforme descrito en Algoritmo 5.3 con cada par de padres seleccionados para generar cada hijo (línea 5, Algoritmo 5.7). Además, para introducir perturbaciones en las soluciones que ayuden a ampliar el espacio de búsqueda, se realiza en cada iteración una mutación aleatoria (descrita en Algoritmo 5.2) a cada individuo nuevamente creado (línea 6, Algoritmo 5.7). A continuación, se aplica la misma de búsqueda local (Algoritmo 5.1) que la aplicada al comienzo del algoritmo para mejorar el *fitness* de la descendencia creada en la generación actual (línea 7, Algoritmo 5.7). Por último, se actualiza la población con la nueva generación de individuos (línea 8) siempre que éstos tengan un coste mejor que los peores individuos de la población actual. Esta evolución será repetida generación tras generación hasta que la condición de parada del algoritmo sea alcanzada (línea 3), que en nuestro caso es una limitación temporal, como se describirá en el siguiente capítulo. Cuando esto

ocurre, la metaheurística devuelve como solución final el mejor individuo alojado en la población (línea 10).

Algoritmo basado en colonia de abejas

El algoritmo basado en colonia de abejas (ABC –*Artificial Bee Colony*) [44-46] es una metaheurística basada en población cuyo funcionamiento se inspira en la inteligencia de las abejas recolectoras de miel. Se distinguen tres tipos de abejas en la colonia, que se corresponden con tres tipos de operaciones de búsqueda de acuerdo con sus comportamientos: abejas obreras, observadoras y exploradoras. La posición de una fuente de alimento representa una solución factible para el problema de optimización que se maneja, en nuestro caso, una planificación de frecuencias válida, y la cantidad de néctar de la fuente de alimento se corresponde con la calidad (el *fitness*) de la solución correspondiente. La solución se codifica como en el resto de casos, de acuerdo con lo explicado anteriormente (Figura 5.1). El esquema general del algoritmo adaptado a la resolución del problema que se maneja en esta tesis se muestra en Algoritmo 5.8.

Algoritmo 5.8 – Pseudocódigo del algoritmo ABC

```
1:  población ← inicializar_población_aleatoria
2:  población ← búsqueda_local (población)
3:  mientras (! límite_tiempo) hacer
4:      población ← mutar_soluciones (abejas_obreras, población)
5:      vectorProb ← generar_probabilidad_soluciones (población)
6:      población ← generar_soluciones (abejas_observ, vectorProb, población)
7:      población ← reemplazar_peores_soluciones (abejas_explor, población)
8:  fin mientras
9:  devolver mejor_solución (población)
```

El algoritmo comienza con la generación aleatoria de la población (línea 1) y la mejora de las soluciones (o planificaciones de frecuencia) que están contenidas en dicha población (línea 2). El método de mejora utilizado es la

búsqueda local descrita en el Algoritmo 5.1. El tamaño de la población representa en este caso el tamaño de la colonia. Cada generación del algoritmo está dividida en cuatro etapas. En la primera trabajan las abejas obreras, que suelen representar hasta la mitad del número de abejas de la colonia completa, y cuyo trabajo consiste en mutar las soluciones que les corresponde de la colonia utilizando el método de mutación aleatorio descrito en el Algoritmo 5.2 (línea 4, Algoritmo 5.8). Esta operación está controlada por un parámetro que indica la probabilidad de mutación. Tras esta operación, las soluciones resultantes son mejoradas utilizando la búsqueda local descrita en Algoritmo 5.1 (línea 4, Algoritmo 5.8). En la segunda etapa se genera un vector de probabilidad, *vectorProb*, que asigna una probabilidad para cada solución dependiendo de la calidad de ésta (línea 5), de tal forma que a mejor *fitness* (coste más bajo), mayor probabilidad (la probabilidad más alta está asignada para la mejor solución de la población). Este vector representa la probabilidad que tiene cada solución de ser explorada por las abejas observadoras. Si las soluciones tienen un *fitness* similar, éstas tienen aproximadamente las mismas opciones de ser exploradas, mientras que las más pobres tienen menos opciones (aunque no ninguna) y las mejores las probabilidades más altas (aunque no la certeza absoluta de ser exploradas). Después de esto, las abejas observadoras generan nuevos individuos a partir de las soluciones de la población y del vector de probabilidad previamente generado (línea 6). En esta ocasión, se aplica un método de mutación muy similar al aplicado anteriormente, en la línea 4, pero en este caso las soluciones no son elegidas al azar, sino seleccionadas de acuerdo con el vector de probabilidad generado en la línea 5. Finalmente, las abejas exploradoras reemplazan las peores soluciones de la población (línea 7) por otras generadas aleatoriamente (y mejoradas mediante el método de búsqueda local explicado en el Algoritmo 5.1). Esta operación permite expandir el espacio de búsqueda del algoritmo, evitando que la metaheurística se estanque en un óptimo local. Al final del proceso, cuando el límite temporal detiene la evolución del algoritmo (línea 3, Algoritmo 5.8), el mejor individuo de la población es devuelto como solución final (línea 9).

Búsqueda local iterativa

La búsqueda local iterativa (ILS –*Iterated Local Search*) [40, 41] es una metaheurística basada en trayectoria con un esquema muy sencillo que basa su funcionamiento en practicar en cada iteración dos operaciones sobre la solución del problema con la que está operando: primero, una operación de perturbación y a continuación, una operación de búsqueda local (de ahí el nombre del algoritmo, que trata de hacer evolucionar una solución mediante la aplicación reiterada de estas dos operaciones). La solución con la que trabaja el algoritmo ILS es una planificación de frecuencias que da solución al problema que se maneja en esta tesis, y que está codificada como en el resto de casos, de acuerdo con la explicación dada anteriormente (Figura 5.1). El Algoritmo 5.9 resume el funcionamiento del citado algoritmo.

Algoritmo 5.9 – Pseudocódigo de la búsqueda local iterativa

```
1:  solución ← generar_solución_inicial
2:  solución ← búsqueda_local (solución)
3:  mientras (! límite_tiempo) hacer
4:      nueva_solución ← mutación_aleatoria (solución)
5:      nueva_solución ← búsqueda_local (nueva_solución)
6:      solución ← actualizar_solución (solución, nueva_solución)
7:  fin mientras
8:  devolver solución
```

Como se puede observar, la heurística está basada en la evolución de una única solución que es generada de manera aleatoria al principio del algoritmo (línea 1). A continuación se aplica el método de búsqueda local explicado en Algoritmo 5.1 (línea 2, Algoritmo 5.9). Después, la solución es perturbada de acuerdo con el Algoritmo 5.2 para evitar que la evolución de ésta se estanque en un mínimo local (línea 4, Algoritmo 5.9). Este proceso está controlado por una probabilidad de mutación, que es el único parámetro a configurar del algoritmo. Tras el citado proceso la solución es mejorada mediante el mismo método de búsqueda local aplicado anteriormente (línea 5). Finalmente, la nueva solución generada es comparada con la que ya se tenía (línea 6), quedándose el

algoritmo con aquella de mejor calidad, es decir, con aquella planificación de frecuencias con un coste menor. La peor solución es descartada y el algoritmo continuará su evolución con la planificación de frecuencias actualizada. Esta solución será devuelta al final del algoritmo (línea 8), cuando éste satisfaga la condición de parada (línea 3).

Búsqueda de entorno variable

La búsqueda de entorno variable (VNS –*Variable Neighbourhood Search*) [38, 39] es una estrategia basada en trayectoria que cambia de entorno (o de vecindario) durante la búsqueda de la solución óptima. Esta característica hace que el algoritmo reduzca las posibilidades de que la evolución de la solución se estanque en un mínimo local. La solución con la que trabaja el algoritmo representa una planificación de frecuencias válida, y está codificada con un vector de enteros, como se ha explicado anteriormente (Figura 5.1). Existen varias versiones del algoritmo VNS, y entre todas ellas, finalmente se incluyó en nuestro estudio la variante sVNS (búsqueda de entorno variable sesgada –*skewed variable neighborhood search*), ya que tras realizar un amplio número de experimentos, era la que mejor resultados obtenía. Su esquema adaptado al problema que se maneja en esta tesis es expuesto en el Algoritmo 5.10.

Algoritmo 5.10 – Pseudocódigo del algoritmo sVNS

```
1:  solución ← generar_solución_inicial
2:  solución ← búsqueda_local (solución)
3:  mientras (! límite_tiempo) hacer
4:      k ← 1
5:      mientras (k <= k_max) hacer
6:          nueva_solución ← mutación_aleatoria (solución, k)
7:          nueva_solución ← búsqueda_local (nueva_solución)
8:          solución ← actualizar_solución (solución, nueva_solución)
9:          k ← actualizar_sesgo (solución, nueva_solución)
10:     fin mientras
11: fin mientras
12: devolver solución
```

Como metaheurística basada en trayectoria que es, el sVNS trabaja con una única solución, y trata de evitar los mínimos locales en su evolución realizando búsquedas lejos de la solución con la que está trabajando en ese momento. El algoritmo comienza igual que el algoritmo ILS: con la generación aleatoria de una planificación de frecuencias (línea 1) a la que se le aplicará un método de mejora (línea 2) mediante el algoritmo de búsqueda local descrito en Algoritmo 5.1. Sin embargo, cada iteración del sVNS depende de un parámetro, *k*, que se utiliza en la perturbación de la solución para controlar la probabilidad de la mutación. De esta forma, el parámetro *k* comienza valiendo 1 (probabilidad de mutación mínima) al principio del proceso (línea 4), e irá cambiando hasta el valor establecido por k_{max}, que en el caso de nuestro estudio será 9, por lo que la probabilidad de mutación variará desde 0.1 hasta 0.9. El operador de perturbación es la mutación aleatoria (línea 6) descrita en el Algoritmo 5.2, pero en este caso, el parámetro *k* será el que determine si el TRX debe mutar o no. Tras este proceso, se aplica la búsqueda local utilizada anteriormente (línea 7) y se escoge para la próxima generación el individuo con mejor *fitness* (aquella planificación de frecuencias con coste menor –línea 8). Finalmente, el parámetro *k* es actualizado (línea 9) teniendo en cuenta la distancia entre la solución nueva y la solución actual y el valor de un nuevo parámetro α que debe ser configurado (línea 9). La distancia entre soluciones es medida contando el número de frecuencias que difieren dentro del mismo sector de ambos individuos, al igual que se hacía en el algoritmo SS. El nuevo valor del parámetro *k* indica si la búsqueda se realizará cerca o lejos de la solución actual en la próxima iteración del algoritmo. Al final del proceso, cuando la evolución de la solución se detiene debido a que se satisface la condición de parada (línea 3), el algoritmo devuelve la mejor planificación de frecuencias que ha sido capaz de calcular (línea 12).

Procedimiento de búsqueda avariciosa, aleatoria y adaptativa

El procedimiento de búsqueda avariciosa, aleatoria y adaptativa (GRASP – *Greedy Randomize Adaptive Search Procedure*) [42, 43] es una metaheurística basada en trayectoria que combina procedimientos constructivos, con los que se genera una solución al problema, y una estrategia de búsqueda local, con la que

se mejora la solución creada en la etapa de construcción. Este proceso se repite durante sucesivas iteraciones para hacer que la solución evolucione. El esquema de funcionamiento del GRASP puede observarse en el Algoritmo 5.11.

Las soluciones avariciosas y aleatorias son generadas añadiendo elementos, en nuestro caso frecuencias, a un conjunto ordenado donde sus elementos son clasificados de acuerdo a una función avariciosa que indica la calidad de las soluciones que se obtendrían utilizando cada elemento del conjunto. Para conseguir variabilidad en la lista de elementos avariciosos, los elementos candidatos clasificados por su calidad son almacenados en una lista restringida de candidatos (RCL –*restricted candidate list*) y de ahí serán seleccionados aleatoriamente en el proceso de construcción de la solución.

Algoritmo 5.11 – Pseudocódigo del algoritmo GRASP

```
1:  solución ← generar_solución_inicial
2:  solución ← búsqueda_local (solución)
3:  mientras (! límite_tiempo) hacer
4:      nueva_solución ← generar_solución_GR (tipo, k, sesgo)
5:      nueva_solución ← búsqueda_local (nueva_solución)
6:      solución ← actualizar_solución (solución, nueva_solución)
7:  fin mientras
8:  devolver solución
```

Como en el caso de las metaheurísticas basadas en trayectoria descritas anteriormente, el GRASP comienza con la generación aleatoria de una planificación de frecuencia (línea 1) seguida por la mejora de esta primera solución a través del método de búsqueda local descrito en el Algoritmo 5.1 (línea 2, Algoritmo 5.11). Tras este proceso inicial, comienza la parte principal del algoritmo con la generación de una solución avariciosa y aleatoria (línea 4), tras lo que se aplicará a la nueva solución generada el método de búsqueda local que fue utilizado anteriormente (línea 5), y se finalizará la iteración con la actualización de la solución que utilizará el algoritmo en la siguiente iteración (línea 6). Para realizar esta actualización se comparan los costes de la mejor solución almacenada hasta entonces y de la nueva solución generada en la iteración actual. La solución con menor coste, que será la planificación de

frecuencias de mayor calidad, será la que permanezca, descartando la de peor calidad.

El algoritmo GRASP posee varios constructores para la generación de la solución avariciosa y aleatoria. El parámetro *tipo* en el pseudocódigo del algoritmo (línea 4) indica el constructor que se aplica en cada caso. De acuerdo con los experimentos realizados, el constructor que proporciona mejores soluciones para el problema abordado en esta tesis es aquel que da nombre a la variante RG del GRASP. RG significa primero aleatorio (*Random*) y luego avaricioso (*Greedy*), y como es la versión que finalmente se empleó, es la que se describirá brevemente en este apartado. El lector que lo desee puede ampliar la información sobre otros constructores del GRASP en las referencias [42, 43].

La primera operación que se realiza cuando se construye una nueva solución consiste en evaluar el coste que supondrá incorporar cada frecuencia que se añade a la solución que se está construyendo. La evaluación de cada elemento que se incorpora llevará a la construcción de una lista de candidatos restringida (RCL), que normalmente se crea con los mejores elementos. En nuestro caso, los mejores elementos son aquellos cuya incorporación a la solución parcial que se está construyendo resulta en un incremento mínimo en el coste de dicha solución. Este es el aspecto avaricioso del constructor. Sin embargo, la variante RG modifica esta forma de conformar la RCL en los siguientes términos: en primer lugar, la RCL se rellena con k frecuencias elegidas aleatoriamente, pero el resto de elementos hasta completar la lista se rellena con frecuencias elegidas de manera avariciosa, seleccionando aquellas que añaden menos coste a la solución en construcción. Después de esto, la frecuencia que finalmente se incorporará a la solución parcial se selecciona de manera aleatoria de la RCL (considerando además el parámetro *sesgo* del algoritmo –línea 4, Algoritmo 5.11). Una vez que se incorpora la frecuencia seleccionada a la planificación en construcción, se actualiza la lista de candidatos y se recalculan los costes incrementales de las frecuencias. Esto es lo que le da a la metaheurística su carácter adaptativo. Este proceso se repite hasta que todos los TRX de la solución tienen asignada una frecuencia.

Aunque las soluciones que se obtienen utilizando el constructor del GRASP son de bastante calidad, siempre son mejoradas aplicando el método de búsqueda local descrito en el Algoritmo 5.1. El proceso de construcción (línea 4,

Algoritmo 5.11), mejora (línea 5) y actualización (línea 6) se realizará hasta que la condición de parada del algoritmo sea satisfecha (línea 3), en cuyo instante se devolverá la mejor solución encontrada hasta la fecha por el algoritmo (línea 8).

5.3. Hiperheurística paralela

Entre los trabajos realizados en esta tesis, se incluye el diseño y desarrollo de una nueva aproximación paralela basada en búsquedas heurísticas para resolver de manera eficiente el problema de la asignación automática de frecuencias. Como se ha descrito en capítulos anteriores, debido a la naturaleza de las metaheurísticas, éstas se adaptan de manera casi directa a múltiples modelos de paralelismo (capítulo 4.4), y son muchos los trabajos publicados donde se utilizan estas técnicas para mejorar el rendimiento de las técnicas metaheurísticas [47-57]. Tras estudiar y realizar pruebas con múltiples opciones, se concluyó que la técnica que se describe en este apartado obtenía unos resultados de mucha calidad aprovechando de la mejor forma posible los medios de los que se disponía para realizar los experimentos. De esta manera, la técnica desarrollada fue una hiperheurística paralela (PHH –*parallel hyper-heuristic*) donde se trató de sacar el máximo partido de las metaheurísticas que se habían desarrollado previamente. En concreto, se desarrollo un equipo paralelo heterogéneo donde se incluyeron las técnicas descritas en la sección anterior de este capítulo (GRASP, ILS, VNS, GA, SS, PBIL y ABC).

Como se explicó en el capítulo introductorio de esta tesis, una hiperheurística puede ser definida como una heurística para seleccionar otras heurísticas [67-70]. En la Figura 2.1 se representó el funcionamiento de estas técnicas. El espacio de búsqueda con el que trabajan es el de las heurísticas (en lugar del de las soluciones). De hecho, las hiperheurísticas se diferencian de las metaheurísticas en que éstas tienen por espacio de búsqueda las soluciones de un problema, mientras que las hiperheurísticas siempre buscan qué heurística será más apropiada en un caso determinado. Por tanto, cuando se utilizan hiper-heurísticas se intenta encontrar el método más apropiado para una situación determinada, más que intentar resolver propiamente el problema, que será

tarea de la metaheurística seleccionada por la hiperheurística. En la Figura 5.2 aparece el diagrama que muestra el funcionamiento de la PHH propuesta.

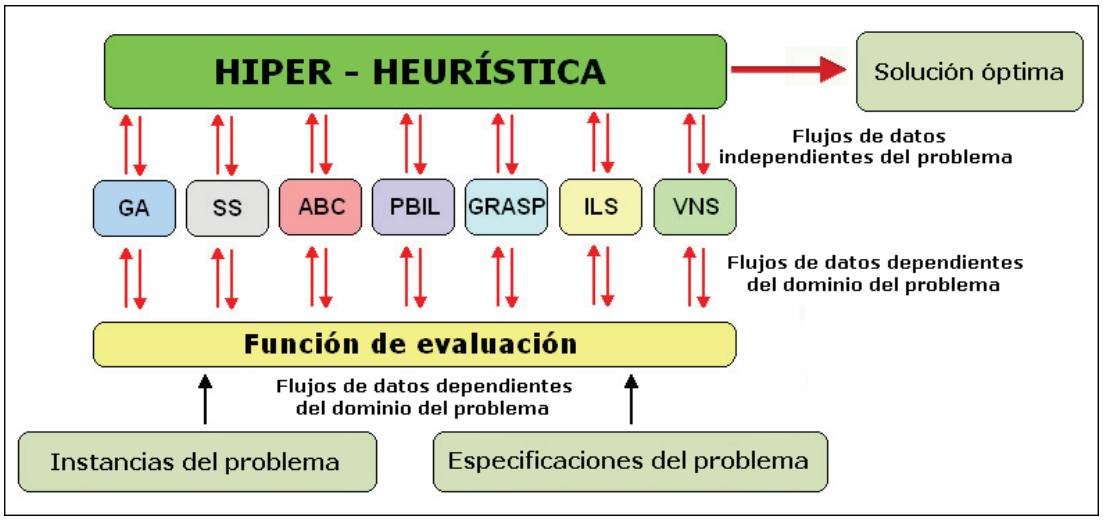

Figura 5.2. Diagrama de funcionamiento de la hiperheurística paralela

Como se puede observar en dicha figura, las encargadas de cargar y gestionar tanto las instancias como las especificaciones del problema son las metaheurísticas, que hacen uso de una función de evaluación especialmente ajustada al problema para evaluar la calidad de las soluciones obtenidas. De hecho, como ya se ha comentado, el problema en sí es resuelto por estas metaheurísticas. La tarea llevada a cabo por el sistema paralelo consiste en controlar la información ofrecida por las metaheurísticas y distribuir la carga de trabajo que cada una realiza teniendo en cuenta la calidad de los resultados que cada una aporta al sistema. La comunicación entre la hiperheurística y las metaheurísticas, que son consideradas las heurísticas de "bajo nivel" del sistema, no depende de las especificaciones del problema, ya que la hiperheurística recibe en cada sincronización del sistema únicamente la siguiente información:

- La mejor solución conseguida por la metaheurística hasta el momento de la sincronización (que como se explicó en la Figura 5.1 está codificada con un *array* de enteros).

- El coste de la solución aportada (que es un dato numérico).

- El identificador de la metaheurística que proporciona dicha solución (que también se codifica por un dato numérico).

Además, la hiperheurística consigue un alto rendimiento en la búsqueda de soluciones óptimas, ya que hace que cada una de las metaheurísticas que integran el sistema realice su búsqueda en paralelo, ampliando en gran medida el espacio de búsqueda de manera altamente eficiente. El sistema desarrollado no selecciona únicamente qué metaheurística se adapta mejor a una situación determinada, sino que determina adecuadamente el número de cores que ejecutarán cada una de las metaheurísticas dependiendo de los resultados que éstas arrojen en cada sincronización del sistema.

Algoritmo 5.12 – Pseudocódigo para el proceso maestro

```
1:  vectorProbHH ← inicializar_vectorProbHH
2:  vectorCores ← asignar_metaheurística_a_cada_core (vectorProbHH)
3:  lanzar_ejecuciones_esclavos (vectorCores)
4:  mientras (! condición_de_parada) hacer
5:     /* El proceso maestro espera a que cada core ejecute su algoritmo */
6:     /* Cada sincronización → los cores envían la mejor solución al maestro */
7:     para (j = 0) hasta (j = número_de_cores_configurados) hacer
8:           vectorSoluciones ← recibir_soluciones_cores (j)
9:     fin para
10:    vectorSoluciones ← ordenar_soluciones_fitness (vectorSoluciones)
11:    vectorProbHH ← actualizar_vectorProbHH (vectorSoluciones)
12:    vectorCores ← actualizar_vectorCores (vectorProbHH)
13:    mejorSolucion ← seleccionar_mejor_solucion (vectorSoluciones)
14:    relanzar_ejecuciones_esclavos (vectorCores, mejorSolucion)
15: fin mientras
16: devolver seleccionar_mejor_solucion_recibida (vectorSoluciones)
```

El sistema paralelo diseñado intenta hacer el mejor uso posible tanto de las metaheurísticas incluidas en él como de los recursos hardware en los que se apoya. Uno de los cores del sistema ejecutará el proceso maestro de la hiper-heurística paralela, mientras que el resto de cores serán utilizados por las metaheurísticas para resolver el problema propuesto. En el Algoritmo 5.12 se puede ver el pseudocódigo del proceso maestro del sistema desarrollado. La hiperheurística paralela se ha diseñado como un sistema síncrono que controla y sincroniza las metaheurísticas que han sido especialmente diseñadas y ajustadas

para resolver el problema que se aborda en esta tesis. Las siete metaheurísticas que integran el sistema se distribuyen entre los cores disponibles, replicándose tantas veces como sea necesario para que todos los cores queden ocupados (línea 2, Algoritmo 5.12). La asignación de metaheurísticas a los cores es controlada por el vector de probabilidad que maneja el sistema (*vectorProbHH,* Algoritmo 5.12). Este vector es inicializado al principio del proceso tal y como se indica en la Figura 5.3 antes de comenzar con la distribución de las metaheurísticas entre los cores que integran el sistema.

Por tanto, se espera que las metaheurísticas queden distribuidas de manera homogénea al principio de la ejecución del algoritmo. Además, para que el sistema no pierda diversidad en su búsqueda, se establece una representación mínima de cada metaheurística en el mismo, de manera que ninguna pueda desaparecer a lo largo de la ejecución del algoritmo. Esta representación variará dependiendo del número de cores sobre los que se ejecute la PHH.

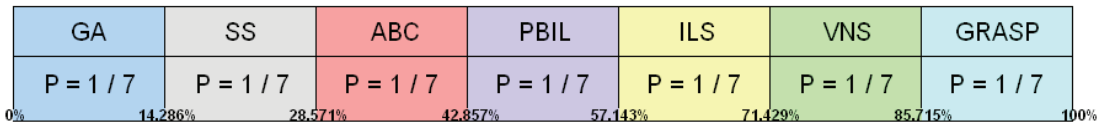

Figura 5.3. Vector de probabilidad inicial de la hiperheurística paralela

Una vez que todos los cores del sistema han sido ocupados por una metaheurística, éstas comenzarán su ejecución tal y como se describe en el Algoritmo 5.13 hasta que se satisfaga el límite temporal que indica que debe realizarse una sincronización con el proceso maestro. Todos los procesos esclavo tienen un esqueleto común (Algoritmo 5.13), que queda establecido por las comunicaciones e inicializaciones necesarias, pero cada metaheurística tiene su esquema de funcionamiento propio, tal y como se describió en los apartados anteriores de esta tesis.

Cuando termina la ejecución de cada metaheurística, éstas enviarán la mejor solución encontrada hasta ese momento al proceso maestro (línea 8 del Algoritmo 5.13 y línea 6 del Algoritmo 5.12). El proceso maestro del sistema debe esperar hasta que las soluciones que son enviadas por cada metaheurística comienzan a llegarle. Cuando esto sucede, el proceso maestro recibe todas las

soluciones (líneas 7 y 8, Algoritmo 5.12) y después las ordena de acuerdo con su calidad (línea 10, Algoritmo 5.12).

Algoritmo 5.13 – Pseudocódigo general para los procesos esclavos

```
1:  Inicializar MPI → MPI_Init
2:  Obtener_comunicador_proc_maestro → MPI_Comm_get_parent
3:  Establecer_tiempo_sincronización (TIEMPO_SINC)
4:  Cargar_datos_metaheurística
5:  Recibir_info_maestro (metaheur_ejecutar, mejor_solución) → MPI_Recv
6:  /* Cada metaheurística → inicializa el número de soluciones que necesite */
7:  /* Ejecución de la metaheurística hasta que TIEMPO_SINC se cumpla*/
8:  Enviar_mejor_solución_maestro (mejor_sol, coste, id_metah) → MPI_Send
9:  Finalizar MPI → MPI_Finalize
```

Después del proceso de recepción, se actualiza el vector de probabilidad del sistema con las mejores soluciones recibidas, siguiendo un método de presión selectiva (línea 11, Algoritmo 5.12). De esta forma, si dentro de la lista de las mejores soluciones se encuentran las cincuenta mejores planificaciones de frecuencias generadas por el sistema, el vector de probabilidad incrementará en 0.02 unidades la probabilidad de la metaheurística correspondiente cada vez que se encuentre una solución generada por ésta dentro de la lista de las mejores candidatas. Así, si por ejemplo en una sincronización la distribución de las cincuenta mejores soluciones es la siguiente: 17 conseguidas por SS, 9 por el GA, 7 por el ABC, 6 por el VNS, 5 por PBIL, 4 por ILS y 2 por GRASP, el vector de probabilidad del sistema actualizado queda como se muestra en la Figura 5.4.

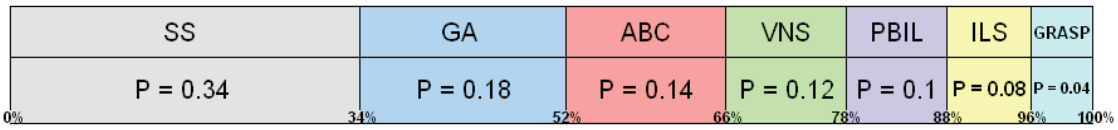

Figura 5.4. Ejemplo de vector de probabilidad actualizado después de una sincronización

Después de la actualización del vector de probabilidad, la hiperheurística utiliza las nuevas probabilidades de este vector para reasignar a cada núcleo de procesamiento del sistema una metaheurística (línea 12, Algoritmo 5.12). Además, la hiperheurística emplea técnicas elitistas a la hora de relanzar las

distintas metaheurísticas, ya que a todas las estrategias se les envía la mejor solución conseguida hasta ese momento antes de relanzar sus ejecuciones. De esta manera, las técnicas basadas en trayectoria (ILS, VNS y GRASP) continuarán la evolución a partir de la solución que se les envía, mientras que las técnicas basadas en población (GA, SS, ABC y PBIL) incluirán en su población inicial la mejor solución conseguida por el sistema hasta entonces. Con esto se conseguirá que la evolución conseguida hasta ese momento por el sistema no sea desechada, continuando cada algoritmo con la búsqueda independiente del óptimo, pero beneficiándose a la vez de los avances conseguidos por todo el sistema paralelo.

El proceso de recepción, actualización y reenvío que se produce en cada sincronización se repetirá hasta que la condición de parada del proceso maestro sea satisfecha (línea 4, Algoritmo 5.12), momento en el cuál la hiperheurística devolverá la mejor solución que se haya conseguido obtener hasta ese momento (línea 16, Algoritmo 5.12), que en el caso del problema que se maneja en esta tesis, será una planificación de frecuencias de muy alta calidad para el problema FAP abordado.

5.4. Resumen

En este capítulo se ha explicado el diseño detallado de las principales técnicas desarrolladas para dar solución eficiente al problema de optimización que se aborda en esta tesis. Las estrategias que finalmente se desarrollaron fueron fruto de un estudio intensivo dentro del dominio del problema y del campo de los métodos heurísticos. Este capítulo se ha dividido en tres grandes partes. En la primera se ha descrito un conjunto de generalidades que fueron aplicadas a todos los algoritmos, como la descripción del individuo utilizado o el algoritmo de búsqueda local que se utilizó en combinación con el resto de metaheurísticas. En la siguiente sección se ha explicado el diseño y ajustes del conjunto de metaheurísticas desarrolladas. En este sentido, se seleccionó un conjunto heterogéneo de técnicas que abordaran la resolución del problema propuesto en esta tesis desde varios enfoques, de manera que la exploración del espacio de

búsqueda fuera lo más diversa y exhaustiva posible. Se han descrito cuatro metaheurísticas basadas en población: PBIL, SS, GA y ABC; y otras tres basadas en trayectoria: ILS, VNS y GRASP, de forma que se ha tratado de cubrir un amplio rango de estas técnicas (población, trayectoria, probabilísticas, colonias de insectos, evolutivos,…). Por último, se ha puesto fin al capítulo con la explicación de un eficiente modelo paralelo que se diseñó y desarrolló con el objetivo de obtener resultados de máxima calidad minimizando el tiempo empleado para ello. Dicho modelo surgió del estudio previo realizado con otras aproximaciones, y fue un equipo heterogéneo de metaheurísticas controlado por una hiperheurística que ejercía de proceso maestro del sistema. Antes de llegar al diseño final de dicho modelo se diseñaron versiones previas del mismo, como el descrito en la publicación [56], y que utilizaba diferentes versiones del algoritmo GRASP sobre un entorno distribuido. Sin embargo se concluyó que utilizar diferentes metaheurísticas sobre un entorno de alto rendimiento producía planificaciones de frecuencia de mayor calidad, por lo que el resultado final fue una técnica a la que llamamos hiperheurística paralela, la cual haciendo uso de las siete metaheurísticas anteriormente mencionadas consiguió resultados muy satisfactorios para el problema abordado en esta tesis.

Capítulo 6

Análisis de resultados

En este capítulo se explican y discuten tanto los principales experimentos realizados con las técnicas descritas en el capítulo anterior, como los resultados obtenidos con éstas. Está dividido en cinco secciones. En la primera se describen las especificaciones tanto hardware como software de los sistemas utilizados para realizar los experimentos. En la segunda se detallan las instancias reales del problema FAP que han sido utilizadas en esta tesis. A continuación se exponen y discuten los experimentos principales llevados a cabo con cada metaheurística, indicando la configuración paramétrica óptima y discutiendo los resultados obtenidos con cada aproximación. En la siguiente sección, se presentan y discuten los resultados obtenidos con la hiperheurística paralela descrita en el capítulo anterior. Finalmente, se presenta un análisis comparativo de todos los resultados obtenidos en nuestro estudio, tanto entre las técnicas desarrolladas en esta tesis como comparando los resultados obtenidos en nuestra investigación con otros publicados en la bibliografía.

6.1. Especificaciones HW y SW de los experimentos

La mayor parte de los experimentos que se han realizado para esta tesis han sido ejecutados en un clúster con 16 nodos idénticos cuyo montaje y configuración se llevaron a cabo en el año 2009. Las características de cada nodo son las siguientes:

- SuperMicro® modelo Superserver 6015B-3/T®

- 2 procesadores Intel® Quad Xeon® CPU E5410 @2,33Ghz, caché L2 6MB
- 8 GB de RAM DDR2
- HD 160Gb

Lo que hace que se disponga de un total de 128 procesadores (16 nodos × 2 procesadores por nodo × 4 núcleos por procesador) y de 128 GB de RAM (distribuida en módulos de 8 GB por nodo). Además, todos los nodos del clúster se encuentran debidamente instalados en un armario rack 19" GESAB® Modelo EGLON Solution V2-Telco, e interconectados mediante un switch de alta velocidad TrendNet® Modelo TEG-240WS 24-Port 10/100/1000Mbps.

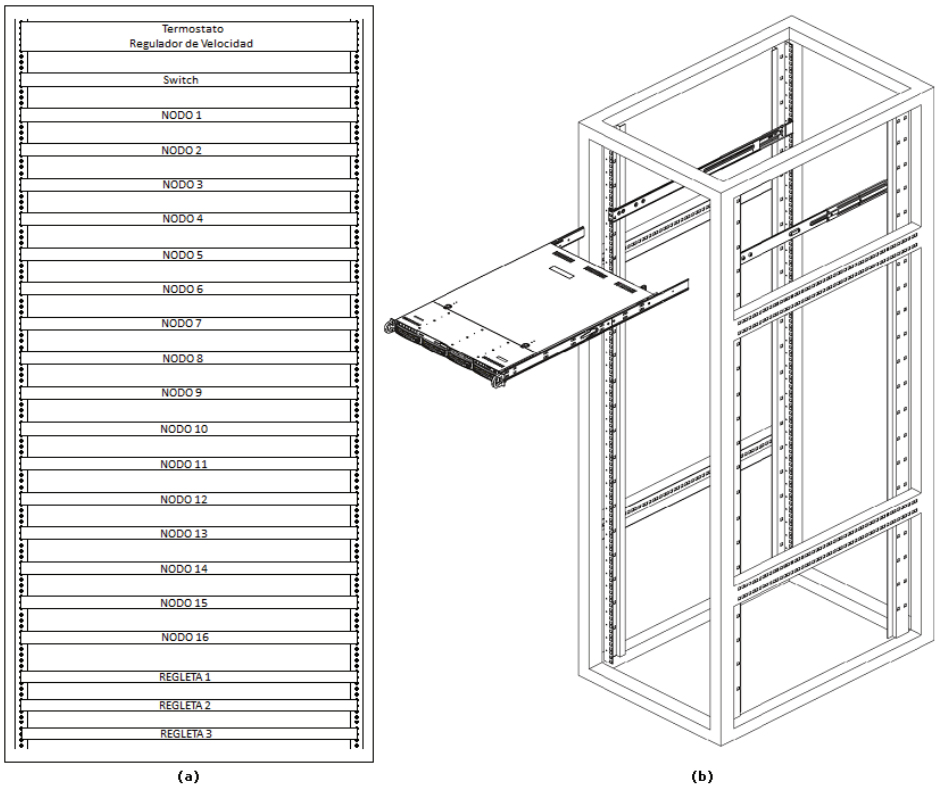

Figura 6.1. Diseño de distribución y montaje de los nodos y los distintos elementos del clúster en su armario rack. (a) Distribución de elementos en el armario. (b) Diagrama de montaje de los nodos mediante sistema de raíles

La Figura 6.1 muestra la distribución de los nodos y de los distintos elementos que se diseñó para el montaje del clúster en el armario rack. El resultado final del montaje puede observarse en la Figura 6.2.

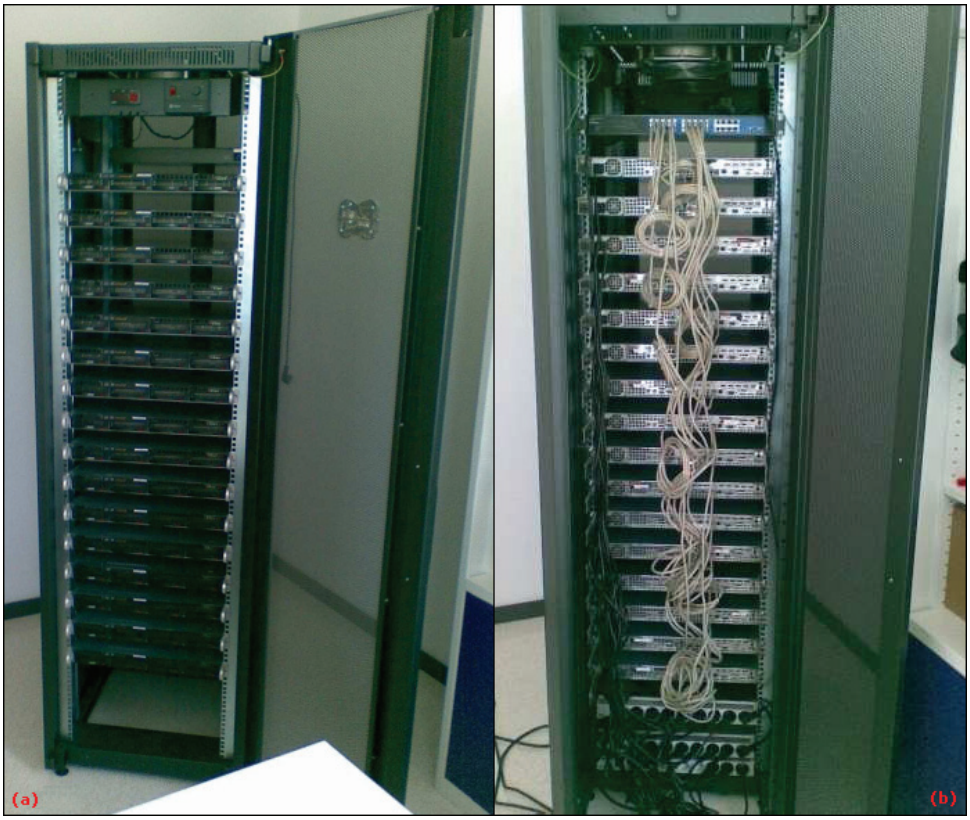

Figura 6.2. Fotografía frontal (a) y trasera (b) del montaje final del clúster

En cuanto a la configuración software del clúster, se llevó a cabo una instalación completa de los programas necesarios en uno de los nodos, realizando una clonación de éste en los restantes nodos. El software instalado y configurado en todos ellos fue el siguiente:

- Sistema Operativo Scientific Linux 5.3 (64 bits) [192]
- Librería MPI (Message Passing Interface) [140]: MPICH2 1.0.8p [193]
- Compilador de C++: GCC 4.1.2 [194]

Otro entorno de trabajo que también fue utilizado para ajustar algunos de los parámetros de los primeros algoritmos desarrollados fue la infraestructura grid. En concreto, se realizaron experimentos utilizando la infraestructura ofrecida por el Ceta-Ciemat [195], y más concretamente, se utilizaron los recursos del proyecto EELA [196], para lo cual se diseñó un sistema maestro-esclavo asíncrono (como el mostrado en la Figura 4.8) por medio de scripts con el que preparar los trabajos de computación (*jobs*) y enviarlos a la grid, ejecutar los correspondientes algoritmos, que se ejecutan en los correspondientes

elementos de computación de la grid en forma de *jobs*, y recibir los resultados de las ejecuciones independientes de los experimentos.

6.2. Instancias del problema FAP abordadas

Aunque en los primeros pasos que se dieron con el problema de la asignación automática de frecuencias se utilizaron instancias de prueba (más concretamente las instancias de Philadelphia [145]), los resultados más interesantes de nuestra investigación se obtuvieron utilizando instancias reales del problema. Por tanto, esta tesis se ha centrado en los experimentos más importantes llevados a cabo sobre dos instancias reales del problema FAP. En concreto, se han proporcionado planificaciones de frecuencias de calidad para dos redes GSM reales que se corresponden con dos ciudades bastante grandes de EE.UU. Las instancias se denominan Seattle y Denver en honor a las ciudades a las que dan servicio, teniendo ambas urbes una población de más de medio millón de habitantes. La Figura 6.3 muestra la topología de ambas instancias, que fueron proporcionadas por investigadores de la Universidad de Málaga gracias a la colaboración dentro del proyecto OPLINK [14]. Cada uno de los triángulos de la figura representa una antena donde que está formada por varios sectores, dos o tres normalmente, donde a su vez hay instalados una serie de TRXs, entre dos y seis, que operan para dar servicio a las comunicaciones producidas en una cierta área de la red. El número y orientación de sectores dentro de la antena dependerán de las zonas de terreno a las que haya que dar cobertura, mientras que el número de TRX dentro de cada sector variará dependiendo de la demanda de tráfico y de los requerimientos específicos que haya en cada zona de la ciudad, si bien el caso más habitual para ambas instancias, Seattle y Denver, son tres o cuatro TRX instalados en cada sector.

Como se ha explicado en los primeros capítulos de esta tesis, la planificación de frecuencias es una tarea compleja muy importante para los operadores GSM actuales que dan cobertura en todos los países del mundo, ya que en todos los casos reales, el rango de frecuencias que están disponibles para dar servicio a cada red de telecomunicaciones es muy reducido.

En el caso de las instancias GSM abordadas en esta tesis, Denver (Figura 6.3) incluye 2612 TRXs (transmisores-receptores) distribuidos en 334 BTSs (antenas) y únicamente tiene disponibles 18 frecuencias para asignar a todos y cada uno de los TRX de la red. Por otro lado, Seattle (Figura 6.3) incluye 970 TRXs instalados en 503 antenas y únicamente 15 frecuencias diferentes para realizar la planificación de frecuencias en toda la red. Por tanto, a la vista de las cifras que se manejan en ambas instancias, queda claro que conseguir planificaciones de frecuencias de alta calidad es una tarea tan importante como compleja.

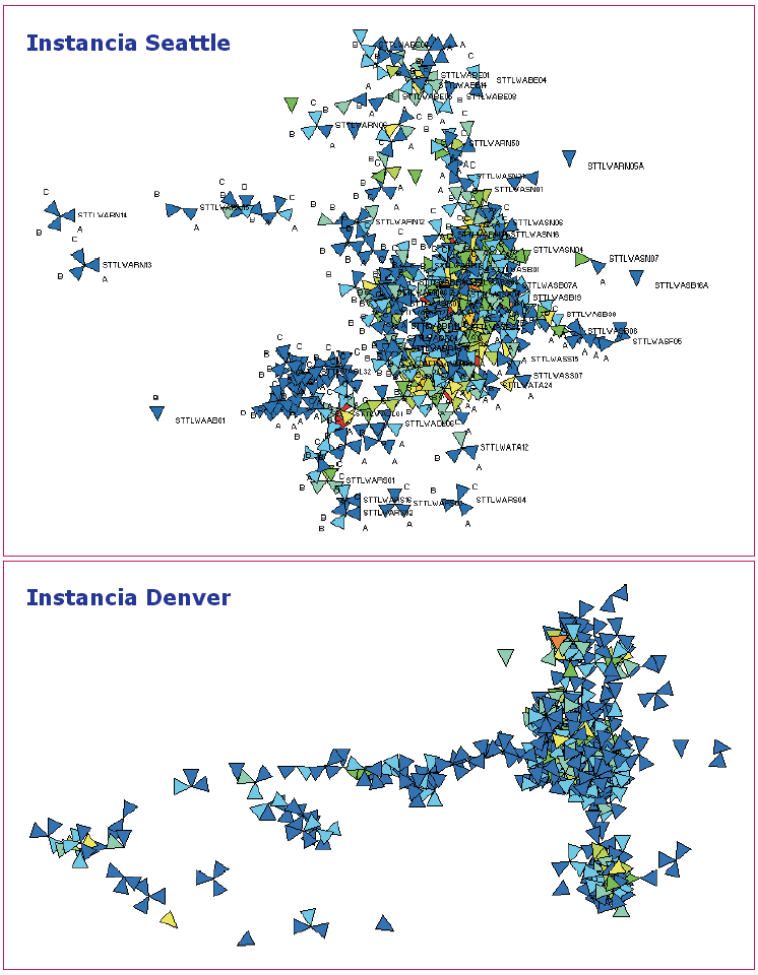

Figura 6.3. Topología de las instancias GSM abordadas

La formulación matemática que se aplica a las instancias anteriormente descritas se explicó en el capítulo 3 de esta tesis, donde se definieron algunas constantes dependientes de las redes con las que se trabajara. Esta información se corresponde con las ecuaciones 3.2, 3.3 y 3.6, y los valores de las constantes

mencionadas son las siguientes: K = 100000, C_{SH} = 6 dB y C_{ACR} = 18 dB, tanto para la instancia Seattle como para la Denver.

Finalmente, se quiere destacar que los datos a partir de los cuales se ha construido la matriz de interferencias que se utilizan todos los algoritmos para extraer la información relativa a las interferencias que se producen entre todo par de TRX de la red (ver Capítulo 3) está basada en una distribución de probabilidad del ratio C/I que está basada en miles de informes de medidas de móviles, llamados MMR (*Mobile Measurement Report*) [148]. Esta información extraída de la industria de la telefonía móvil es mucho más precisa que, por ejemplo, los modelos de predicción de propagación, ya que los MMR capturan los patrones de las localizaciones de llamada en la red, y no dependen de las predicciones. Estas propiedades hacen que el problema abordado en esta tesis sea más realista que los basados en *benchmark* que están disponibles en la bibliografía [145], como son el conjunto de instancias de Philadelphia, con el que se realizaron los primeros experimentos de nuestra investigación, o las instancias CELAR, CALMA, GRAPH o COST 259 [145].

6.3. Experimentos y análisis de resultados obtenidos con las metaheurísticas

Debido a la naturaleza estocástica de las metaheurísticas, todos los experimentos realizados en el estudio presentado en esta tesis han sido repetidos de manera independiente 30 veces, con el fin de validar estadísticamente los resultados obtenidos de manera empírica. Además, como se explicó en el capítulo 4.3 de *Evaluación estadística de las metaheurísticas*, se ha realizado un riguroso análisis estadístico de los datos obtenidos con los ajustes de los distintos algoritmos, y como se puede comprobar en el Apéndice I de esta tesis, se demuestra que los resultados obtenidos son estadísticamente relevantes. Por otro lado, para detectar diferencias en la ejecución de los algoritmos en diferentes periodos de tiempo, se han considerado varios límites temporales: 120, 600, 1800 y en ocasiones hasta 3600 segundos. En este sentido, no se han tenido en cuenta tiempos inferiores a dos minutos porque,

debido a la complejidad del problema, en dichos casos las metaheurísticas tendrían demasiado poco tiempo para hacer evolucionar las soluciones, por lo que los resultados no serían significativos. Por otro lado, tampoco se han considerado tiempos que en la mayoría de los casos superaran los treinta minutos (de manera excepcional una hora), ya que éstos serían excesivos para un problema real de ingeniería, que requiere soluciones de la máxima calidad posible en tiempos razonables. A esta razón hay que sumarle que en la bibliografía hay publicados importantes trabajos con otras técnicas diferentes a las presentadas en esta tesis que emplean los mismos límites temporales, por lo que también se consideraron para facilitar la ejecución de estudios comparativos con dichos estudios.

Hemos de decir que se han realizado cientos de experimentos con cada algoritmo para ajustar sus diferentes parámetros de manera que se obtuvieran los mejores resultados posibles para el problema descrito en esta tesis. En esta sección se resumen y analizan los resultados obtenidos por las propuestas algorítmicas descritas en el capítulo anterior. Sin embargo, antes de analizar dichos resultados, conviene explicar su representación. Como se describió en la Sección 5.1, la forma de codificar las soluciones es mediante *arrays* de números enteros, p, donde $p(t_i) \in F$ es la frecuencia que se asigna al TRX t_i de la red (Figura 5.1). Una planificación de frecuencias completa está formada por dicha asignación para cada uno de los transmisores de la red que se esté abordando (Sección 6.2). Cada una de estas planificaciones tiene un coste asociado que determina su calidad. Este coste es un número decimal que representa las unidades de coste asociadas a las interferencias que se generan en la red para la planificación evaluada.

Tabla 6.1. Costes medios ($\bar{x}$) y desviaciones estándar (σ) de las interferencias producidas en 30 planificaciones de frecuencias generadas aleatoriamente para las instancias Seattle y Denver

	Planificaciones de frecuencia aleatorias $\bar{x} \pm \sigma$
Seattle	55204,11 ± 1815.99
Denver	115577436,48 ± 3902668.53

Como ya se ha mencionado, la versión del problema FAP en la que se integran las instancias GSM reales que se abordan en esta tesis es la MI-FAP (*Minimum Interference FAP*), que consiste en minimizar las interferencias que se producen en la red generando una planificación de frecuencias de la máxima calidad que sea posible. Por tanto, el objetivo a minimizar es el coste de las planificaciones que se obtienen con cualquiera de los algoritmos utilizados. Una solución con un coste mínimo será una solución de máxima calidad, que es el objetivo del problema abordado en esta tesis. En la Tabla 6.1 se muestra la media ($\bar{x}$) y la desviación estándar (σ) del coste asociado a 30 planificaciones de frecuencia generadas de manera aleatoria.

Como se puede observar, el coste asociado a planificaciones de frecuencia generadas aleatoriamente es de miles de unidades de coste en caso de la instancia Seattle y de millones en caso de la instancia Denver. En las siguientes secciones se expondrán los resultados conseguidos por los algoritmos desarrollados para esta tesis. La primera conclusión que se puede extraer es que las técnicas propuestas reducen varios órdenes de magnitud los costes de las soluciones generadas aleatoriamente. Esta reducción es muy significativa e importante, ya que ninguna comunicación será posible en una red que tuviera las interferencias con los costes que se muestran en la Tabla 6.1.

Experimentos y resultados con la búsqueda dispersa

La búsqueda dispersa (SS) fue uno de los primeros algoritmos con los que se planteó la resolución del problema FAP abordado en esta tesis. También ha sido el algoritmo con el que más se ha trabajado y más experimentos se han realizado ya que fue utilizado en la realización de trabajos conjuntos con otros investigadores [120, 122]. De hecho, el resto de metaheurísticas se diseñaron teniendo en cuenta consideraciones que se descubrieron o establecieron para la búsqueda dispersa, por lo que el número de experimentos de los sucesivos algoritmos fue bastante menor.

SS fue desarrollado en varias etapas, incluyendo una gran variedad de modificaciones, aunque siempre se ha seguido el esquema descrito en el Algoritmo 5.6. De esta forma, se realizaron pruebas con el algoritmo estándar para ajustar los parámetros básicos (población, *RefSet*, métodos de combinación de soluciones, etc.), pero luego se realizaron diferentes mejoras que resultaron significativas y se volvieron a realizar experimentos para ajustar los parámetros básicos. Por ejemplo, el diseño de búsqueda local para mejorar las soluciones generadas por la metaheurística fue mejorando paulatinamente, si bien, finalmente se fijó la LS descrita en el Algoritmo 5.1 como definitiva, ya que era la que mejores resultados ofrecía. También se hicieron experimentos con diferentes métodos de combinación de soluciones (Algoritmo 5.6), diferentes distancias para la generación del conjunto diverso, diferentes tamaños de población, diferentes tamaños del conjunto de referencia (*RefSet*), diferentes proporciones de calidad y diversidad en el *RefSet*, etc. En definitiva, se han realizado cientos de experimentos para establecer la mejor configuración posible del algoritmo a la hora de resolver el problema FAP en el que se ha aplicado. La Figura 6.4 muestra un resumen de las evoluciones más importantes en los resultados obtenidos por el algoritmo desde uno de los primeros trabajos publicados con él en la resolución del FAP abordado en esta tesis [120] hasta la versión final actual que se presenta en este documento.

En la versión que catalogaremos como inicial, el resultado medio tras 30 ejecuciones independientes transcurridos 30 minutos era de 93820.4 unidades de coste, con una población inicial de 40 individuos, un tamaño del *RefSet* igual a 10 (mitad calidad y mitad diversidad) y con cruce uniforme a nivel de sector como método de combinación de padres. Este resultado de por sí es bastante bueno, ya que como se puede observar en la Tabla 6.1 (fila Denver), rebaja en varios órdenes de magnitud el resultado obtenido de manera aleatoria para la instancia Denver del problema. Sin embargo, el objetivo de esta tesis ha sido desarrollar un conjunto de algoritmos que obtuviera resultados de la mayor calidad posible en la resolución del problema abordado, por lo que las planificaciones de frecuencia generadas en ese punto se tomaron como resultados de referencia para realizar versiones mejoradas del algoritmo. En la gráfica de la Figura 6.4 se ha representado con 5 líneas las 5 mejoras más significativas del algoritmo en su proceso de optimización.

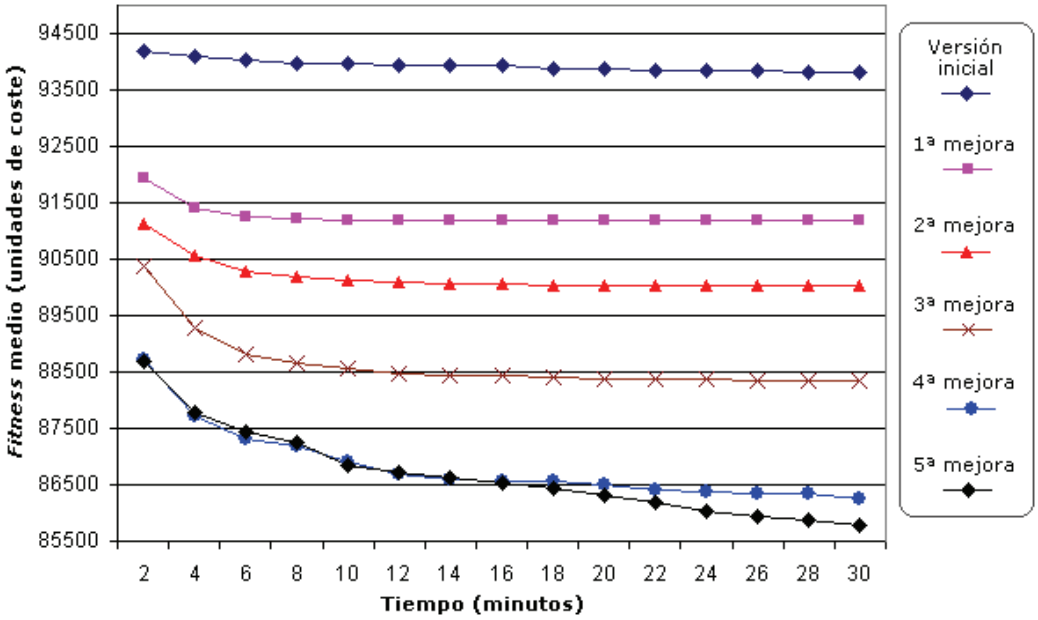

Figura 6.4. Resultados medios de la primera versión publicada y las sucesivas mejoras aplicadas al algoritmo SS en la resolución de la instancia Denver

La primera mejora significativa que se realizó, la cual se corresponde con la línea de cuadros etiquetada como "1ª mejora" en el gráfico de la Figura 6.4, fue producida en gran medida gracias al estudio que se llevó a cabo acerca del cruce de las soluciones. En la versión original del algoritmo (representada por la línea de rombos etiquetada como "Versión inicial" en la Figura 6.4), SS estaba codificado con un cruce uniforme a nivel de sectores (cada 20-40 sectores), y tras la realización del estudio se pudo concluir que realizar el cruce a nivel de TRX (tal y como se explica en el Algoritmo 5.3) resultaba mucho mejor, ya que los resultados mejoraban significativamente. Además, se realizaron experimentos para determinar la frecuencia en la que la operación de cruce debía ser realizada en la creación de una nueva solución. La conclusión final de este estudio fue que el cruce uniforme (mejor que cualquier otra variante de cruce aleatorio) a nivel de TRX era el que mejores resultados ofrecía. De hecho, tal y como se aprecia en la Figura 6.4, la mejora proporcionada es bastante significativa. Este hecho puede explicarse debido a que esta configuración para el cruce genera soluciones de muy diversa índole dependiendo de los padres, lo cual es muy significativo, ya que amplía el espacio de búsqueda abarcado por el algoritmo, lo que se traduce en que las planificaciones de frecuencia que se generan sean de mayor calidad. Así, el coste medio (tras 30 ejecuciones

independientes) de las soluciones generadas tras 1800 segundos de ejecución se redujo de 93820 a 91186, lo cual representa una mejora bastante significativa.

Como se ha comentado, el resultado que tomaremos como base en esta sección se corresponde con la línea "Versión inicial" de la Figura 6.4. Dicho resultado fue el publicado en [120], y aparece expresado en la fila "SS v. 1.0" de la Tabla 6.2.

Tabla 6.2. Resultados empíricos en unidades de coste para tres límites temporales de la versión inicial y final del algoritmo SS en la resolución de la instancia Denver

	120 segundos		600 segundos		1800 segundos	
	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$
SS v. 1.0 [120]	91216.7	94199.6±1172.3	91069.8	93953.9±1178.6	91069.8	93820.4±1192.3
SS v. final [125]	86169.4	88692.7±1124.9	84570.6	86843.8±950.5	84234.5	85767.6±686.3

La siguiente mejora realizada al algoritmo, representada en el gráfico de la Figura 6.4 por la línea de triángulos etiquetada como "2ª mejora", fue fruto del estudio llevado a cabo en el método de generación de conjuntos para crear el *RefSet* al comienzo del algoritmo (ver Algoritmo 5.6). Este método es muy importante, ya que a partir de él se genera el conjunto de individuos con el que SS trabajará y, como regla general, se puede afirmar que cuanto mayor sea la calidad de las soluciones con las que trabaje el algoritmo desde el principio, mejores serán los resultados que se obtengan al final de la ejecución del mismo. De hecho, tal y como se ve en el gráfico de la Figura 6.4, la diferencia entre la versión de la 1ª mejora y de la 2ª mejora de SS está justo en los primeros segundos de ejecución del mismo. Al final, esta mejora redunda en bajar el coste medio de las planificaciones de frecuencia de 91186 a 90032 unidades de coste.

La siguiente mejora en el coste de las soluciones proporcionadas con SS (línea de aspas etiquetada como "3ª mejora" en el gráfico de la Figura 6.4) se produjo gracias al estudio llevado a cabo con las proporciones del *RefSet*. El conjunto de referencia incluye soluciones tanto de calidad como relevantes por su diversidad. Según la bibliografía, lo normal es que este conjunto se encuentre equilibrado, encontrándose en él un número similar de soluciones de calidad y diversas. Este era el caso en las primeras versiones desarrolladas del algoritmo,

de forma que si el *RefSet* contiene 10 soluciones, 5 eran planificaciones de frecuencia de alta calidad y otras 5 constituían soluciones diversas. Sin embargo, tras realizar un estudio donde se experimentó con diversas configuraciones en la distribución de las 10 soluciones que se hallaban dentro del *RefSet* ([1, 9], [2, 8], [3, 7],…, [8, 2], [9, 1]), se concluyó que con una única solución que preservara la planificación de frecuencias de más alta calidad en el *RefSet* era suficiente, y que cuanto más soluciones diversas hubiera, mejores resultados obtenía el algoritmo. Así, la proporción del conjunto de referencia quedó establecida en 1 solución de alta calidad y 9 soluciones diversas (*RefSet* = 10, [1, 9]). La mejora que se produjo con esta modificación fue significativa, ya que en 30 minutos se pasó de 90032 unidades de coste a 88330 (como media de 30 ejecuciones independientes). La conclusión que se puede sacar de estos experimentos es que cuanta mayor diversidad haya en las soluciones con las que se trabaja, mayor es el espacio de búsqueda que se explora, y por tanto, mejores resultados se obtiene. Para mantener la calidad de las soluciones que se generan en siguientes generaciones, es suficiente con mantener el mejor resultado conseguido hasta ese momento.

La cuarta evolución, que se representa por la línea de círculos etiquetada como "4ª mejora" en el gráfico de la Figura 6.4, fue una de las más significativas, ya que mejora la evolución del algoritmo en todas las rodajas de tiempo del mismo, desde el primer minuto de ejecución hasta la finalización de la ejecución del mismo. De hecho, se consigue reducir el coste medio de las planificaciones de frecuencia generadas tras 30 ejecuciones independientes hasta 86263 unidades de coste. Esta evolución se consiguió gracias a una mejora en el método de actualización del *RefSet* al final de cada iteración del algoritmo (ver Algoritmo 5.6). Según el esquema genérico del algoritmo, el *RefSet* debe ser actualizado en cada iteración a partir de la población que se genera al principio de la ejecución del mismo. Tras realizar un amplio conjunto de experimentos, se llegó a la conclusión de que lo mejor era que el subconjunto diverso dentro del *RefSet* no se actualizase de la población inicial, sino que en cada iteración del algoritmo se regenerase el conjunto diverso de manera aleatoria (tras lo cual se aplicaría el método de búsqueda local explicado en Algoritmo 5.1). Esto produce un gran avance en el algoritmo que se traduce en mejores planificaciones de frecuencia generadas en cualquier instante de ejecución del mismo.

Finalmente, la evolución de la última etapa en el estudio con la búsqueda dispersa está representada en la Figura 6.4 con la línea de rombos etiquetada como "5ª mejora". La última versión del algoritmo consigue una buena evolución en los resultados tanto al principio de la ejecución del mismo como al final. La mejora producida en esta etapa fue producida al estudiar de nuevo, tras implantar todas las mejoras descritas anteriormente, el tamaño del conjunto de referencia del algoritmo (el cual resulta una pieza clave en su funcionamiento). Se hicieron de nuevo experimentos variando el tamaño de este conjunto desde 3 soluciones hasta 15, manteniendo las proporciones de una única solución en el subconjunto de calidad dentro de calidad y el resto hasta copar el tamaño del *RefSet* con soluciones de alta diversidad. El resultado final de este estudio estableció que el *RefSet* debía tener un tamaño de 9 soluciones, con las proporciones [1, 8]. Este cambio reducía un 10% el trabajo que el algoritmo debe hacer en cada iteración, haciendo posible que se ejecuten más iteraciones sin reducir significativamente el potencial de la búsqueda que se realiza, con lo que los resultados se mejoraron. El resultado final que se obtuvo tras la incorporación de esta mejora fue publicado en [125], y aparece expresado tanto en la fila "SS v. final" de la Tabla 6.2, así como en las Tablas 6.4 y 6.5 de la sección de resumen de los resultados. De igual forma, el algoritmo ajustado fue también utilizado para resolver la instancia Seattle, arrojando los resultados que se pueden ver en las Tablas 6.6 y 6.7 de la misma sección.

En conclusión, como resumen de los experimentos realizados para las instancias abordadas (Sección 6.2) y tras realizar el pertinente análisis de los resultados obtenidos, se ha determinado que la mejor configuración paramétrica de la búsqueda dispersa para resolver el problema de la asignación automática de frecuencias abordado en esta tesis es la siguiente:

- Tamaño de la población = 40 individuos
- Tamaño del conjunto de referencia = 9 soluciones
- Proporción en el conjunto de referencia = [1, 8]. 1 solución de calidad (la mejor) y 8 destinadas a almacenar soluciones diversas
- Método de combinación de soluciones = cruce uniforme a nivel de TRX
- Método de generación de subconjuntos = selección de pares
- Política de reemplazo = sustitución de los peores individuos

Experimentos y resultados con el aprendizaje incremental basado en población

El algoritmo PBIL fue la primera metaheurística abordada en nuestra investigación. De hecho, los primeros experimentos se llevaron a cabo con instancias tipo *benchmark*, y más concretamente con el conjunto de instancias Philadelphia [145]. Fruto de dicho estudio se obtuvieron algunas publicaciones [129, 197], si bien en una etapa posterior se estudió y analizó el uso de PBIL aplicado al problema FAP en el que se centra la investigación de esta tesis (FAP con instancias reales).

De acuerdo con el esquema del algoritmo, descrito en Algoritmo 5.4, PBIL presenta 4 parámetros principales, que son el tamaño de la población, la probabilidad y la cantidad de mutación y la tasa de aprendizaje. Pero además, se hicieron experimentos con distintas versiones de PBIL (ver descripción del algoritmo en el capítulo 4.2) y con varias configuraciones que añadían estrategias de elitismo al algoritmo.

La configuración inicial de la que se partió vino establecida por los valores de la configuración estándar del algoritmo según su creador [37], que son los siguientes: tamaño de la población = 100 individuos, tasa de aprendizaje = 0.1, probabilidad de mutación = 0.02 y cantidad de mutación = 0.05. El primer parámetro con el que se experimentó, dejando el resto con su valor estándar, fue el tamaño de la población. Según nuestra experiencia previa, tamaños de poblaciones muy grandes no funcionan demasiado bien con instancias grandes de problemas de optimización, por lo que se hicieron experimentos considerando hasta 100 individuos en la población, y se ajustó el parámetro en dos tandas. En la primera se tomaron valores desde 20 hasta 100, y en el segundo se acotó el tamaño de la población de manera más fina de acuerdo al mejor valor conseguido en la primera tanda de experimentos. Los resultados medios de las pruebas llevadas a cabo con la población de PBIL pueden observarse en la Figura 6.5, donde se puede comprobar que el valor óptimo para la población es de 20 individuos, ya que para valores mayores se obtienen peores resultados de media en cualquier franja de tiempo (Figura 6.5.a y Figura 6.5.b). Para valores menores que 20 (Figura 6.5.c), las planificaciones de frecuencia que se obtienen

en los primeros minutos de ejecución son de mayor calidad cuanto más pequeña es la población, ya que el algoritmo es capaz de evolucionar de manera más rápida, pero a partir de 20 minutos de ejecución, los mejores resultados se obtienen con una configuración de 20 individuos en la población, que es finalmente el valor que se estableció para este parámetro.

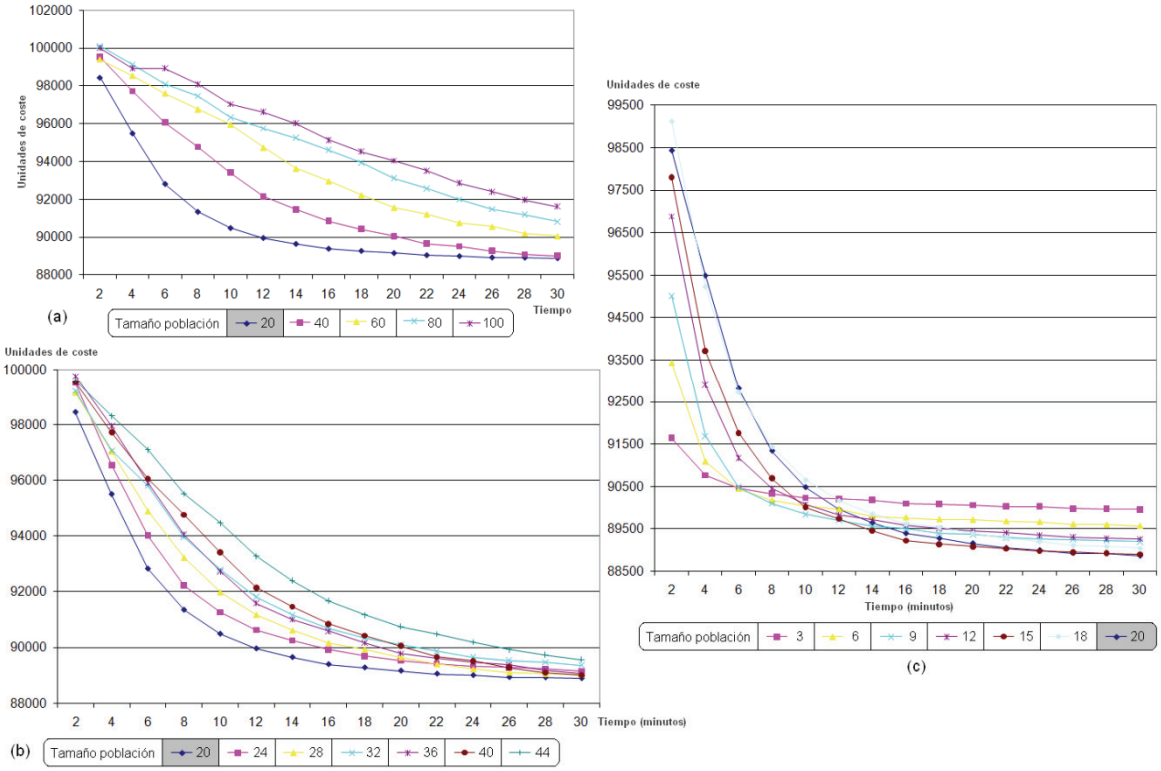

Figura 6.5. Resultados medios de los experimentos llevados a cabo con el tamaño de la población de PBIL para la instancia Denver

Una vez establecido el tamaño de la población, el siguiente parámetro a estudiar fue la tasa de aprendizaje (LR), ya que éste es una parámetro muy importante, pues determina cuánto aprenderá el algoritmo en cada generación de la mejor solución de la población (más concretamente cuánto influirá el mejor individuo en la modificación del vector de probabilidad que representa la población de individuos, como se explica en el Algoritmo 5.4). Se realizaron experimentos con tasas de aprendizaje desde 0.1 hasta 0.9, siendo los resultados los que aparecen en la Figura 6.6.

Como se puede observar, la mejor configuración para la tasa de aprendizaje es 0.1. Tanto valores mayores (Figura 6.6.a) como valores menores (Figura 6.6.b) hacen que el algoritmo funcione peor, ya que en el primer caso se toma demasiada información del mejor individuo de la población, reduciéndose en mucho el espacio de búsqueda, y en el segundo, se tiene en cuenta demasiada poca información, haciendo que la evolución del algoritmo sea mucho más lenta.

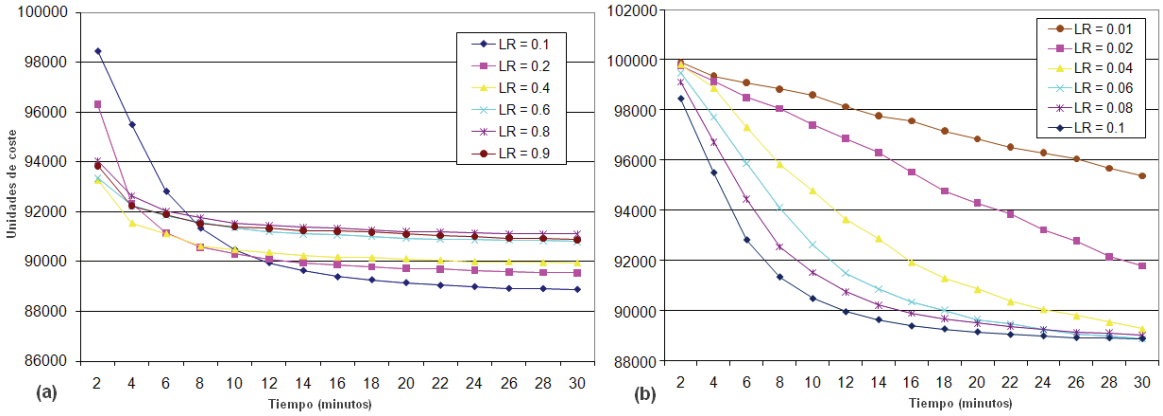

Figura 6.6. Resultados medios de los experimentos llevados a cabo con la tasa de aprendizaje (LR) del algoritmo PBIL para la instancia Denver

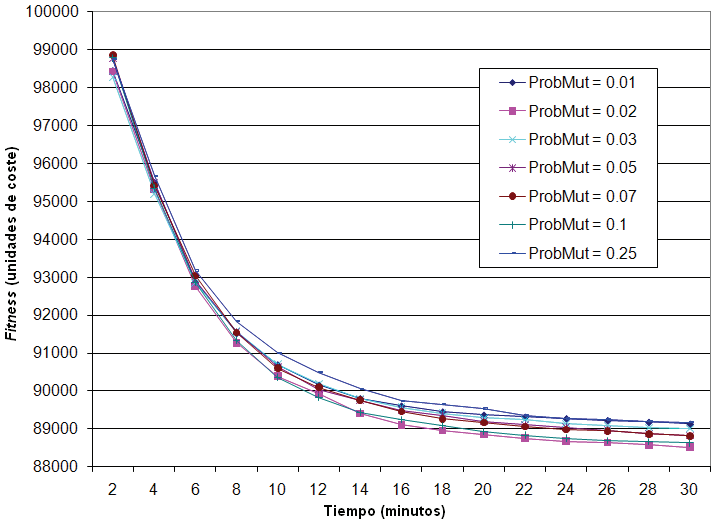

Figura 6.7. Resultados medios de los experimentos llevados a cabo con la probabilidad de mutación (ProbMut) del algoritmo PBIL para la instancia Denver

Con los valores establecidos del tamaño de la población y la tasa de aprendizaje, el siguiente parámetro que se ajustó fue la probabilidad de mutación (Figura 6.7). En este caso, el valor considerado estándar para este parámetro (ProbMut = 0.02) es el que mejores resultados ofrece, ya que tanto para valores inferiores como superiores, los resultados medios que se obtienen son peores, si bien no se presentan diferencias demasiado significativas. En cualquier caso, queda claro que el algoritmo funciona mejor con pocas posibilidades de que el vector de probabilidad que representa la tendencia de la mejor solución de la población cambie. Finalmente, el último parámetro del algoritmo estándar que se sometió a experimentación fue la cantidad de mutación (MutA), que indica el grado de alteración que sufre el vector de probabilidad en caso de que la operación de mutación se lleve a cabo. El valor estándar para este parámetro es de 0.1, y se realizaron pruebas desde 0.05, que representa una muy ligera alteración, hasta 0.75, que representa una modificación del vector de probabilidad muy notoria. El resumen de los resultados que se obtuvieron pueden observarse en la Figura 6.8.

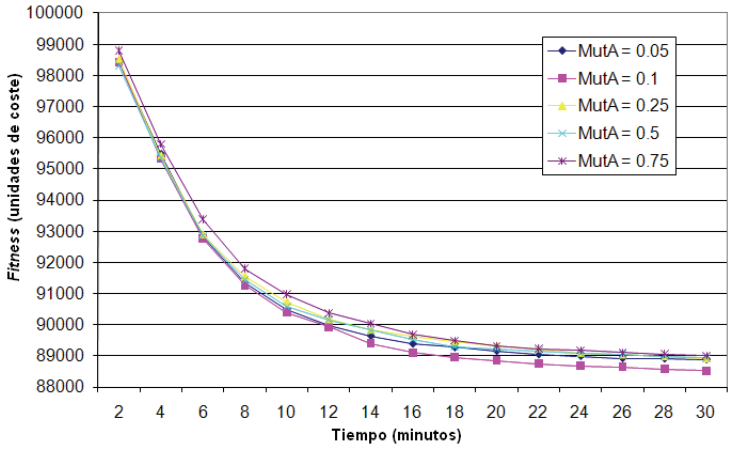

Figura 6.8. Resultados medios de los experimentos llevados a cabo con la cantidad de mutación (MutA) del algoritmo PBIL para la instancia Denver

Como puede comprobarse, no se observan cambios muy notorios en los resultados para distintos valores de este parámetro. Este hecho se debe a que el resto de parámetros se encuentran debidamente ajustados y que además la probabilidad de mutación es muy baja, con un valor de 0.02, por lo que la cantidad de mutación no resulta un factor crítico. Sin embargo, se observa que las mejores planificaciones de frecuencias se obtienen con una cantidad de

mutación igual a 0.1, hecho que resulta más relevante a partir de los 15 minutos de ejecución, por lo que éste fue el valor que finalmente se adoptó para este parámetro.

Por último, se hicieron experimentos con diversas variantes del algoritmo, como son las versiones de PBIL con aprendizaje negativo, equitativo, consenso o PBIL complemento [36, 37]. Sin embargo, los resultados que se obtuvieron probaron que todas las variantes ofrecían resultados más pobres que la versión estándar del algoritmo [128], por lo que ésta fue la versión que finalmente se adoptó en la resolución del problema abordado en esta tesis.

Una vez ajustado el algoritmo con la instancia Denver, se aplicó el mismo, con la misma configuración, a la instancia Seattle, obteniéndose los resultados que se resumen en las Tablas 6.4 y 6.5 para la instancia Denver, y las Tablas 6.6 y 6.7 para la instancia Seattle.

En conclusión, tras la realización de los experimentos para las instancias abordadas (Sección 6.2) y después de realizar el pertinente análisis de los mismos, se ha determinado que la mejor configuración paramétrica del aprendizaje incremental basado en población para resolver el problema de la asignación automática de frecuencias abordado en esta tesis es la siguiente:

- Tamaño de la población = 20 individuos
- Tasa de aprendizaje (LR) = 0.1
- Probabilidad de mutación = 0.02
- Cantidad de mutación = 0.1
- Versión de PBIL = Estándar

Experimentos y resultados con el algoritmo genético

Este clásico algoritmo evolutivo es una de las metaheurísticas más utilizadas en la resolución de problemas de optimización, y la razón se ve confirmada en el estudio llevado a cabo en esta tesis, ya que junto con la búsqueda dispersa, es

la técnica que mejores resultados ofrece en la resolución del problema abordado en esta tesis. De hecho, como se puede ver en las Tablas 6.4 y 6.5 para la instancia Denver y 6.6 y 6.7 para la instancia Seattle, el GA obtiene los segundos resultados más competitivos en la resolución de la instancia Denver y los mejores resultados en la resolución de la instancia Seattle. Una de las razones para su éxito en la resolución del FAP fue el preciso ajuste paramétrico al que fue sometido. Como se puede comprobar en la sección 5.2 (Algoritmo 5.7), el GA presenta bastantes parámetros, que deben ser configurados adecuadamente para que el algoritmo ofrezca su mejor rendimiento. Además, este algoritmo se desarrolló tras PBIL y SS, por lo que muchos detalles de diseño y desarrollo se habían estudiado previamente en los citados algoritmos, y se volcaron en el desarrollo del GA, redundando en un mejor funcionamiento de éste. Así, el tipo de cruce o el operador de mutación utilizado fueron importados de la búsqueda dispersa, ya que los experimentos realizados con ese algoritmo demostraron que tanto el cruce uniforme a nivel de TRX (Algoritmo 5.3), como la mutación aleatoria (Algoritmo 5.2) y también la sustitución de las peores soluciones de la población como política de reemplazo, eran las estrategias más adecuadas para esos operadores genéticos en la resolución del problema abordado en esta tesis.

Por tanto, los experimentos que se realizaron con el GA fueron relativos al tamaño de la población, a la política de selección de padres, al número de cruces (y por tanto de descendientes) que se realizan en cada generación del algoritmo y a la probabilidad de mutación del algoritmo. De acuerdo con nuestra experiencia previa, los valores iniciales que se adoptaron para dichos parámetros fueron: 10 para el tamaño de la población, selección aleatoria de padres, un único cruce por generación (luego un solo descendiente en cada generación, que sustituye a la solución de peor calidad en la población), y 0.2 para la probabilidad de mutación.

El primer parámetro que se configuró fue el método de selección de padres. Se hicieron pruebas seleccionando un par de padres de manera aleatoria, un padre de manera aleatoria y otro de manera avariciosa, los dos padres de manera avariciosa y por último un torneo binario para seleccionar los padres. La Figura 6.9 resume los resultados obtenidos en los experimentos realizados.

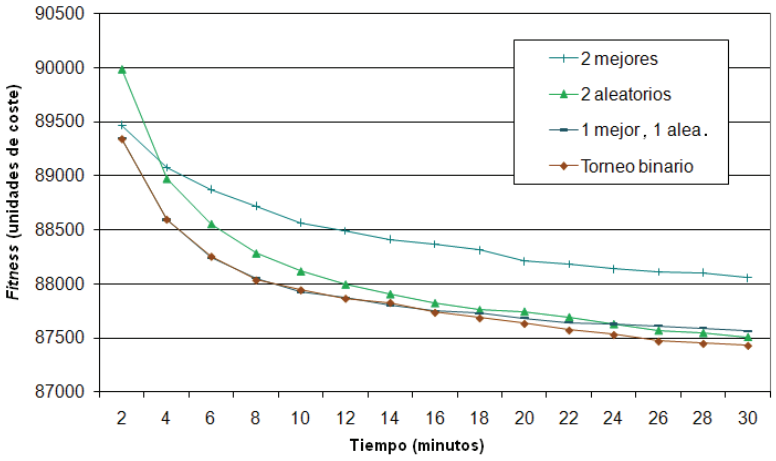

Figura 6.9. Resultados medios de los experimentos llevados a cabo con distintos métodos de selección de padres para resolver con el GA la instancia Denver

Como se puede ver, la técnica más avariciosa, que consiste en seleccionar los dos mejores padres de la población para cruzarlos, obtiene resultados notablemente peores que el resto de técnicas. Para ejecuciones cortas, los métodos que mejor funcionan son el torneo binario y la selección de un padre aleatoriamente y el otro de manera avariciosa, sin embargo, para ejecuciones más largas el torneo binario se diferenció como el mejor método de selección de padres, por lo que es el que se adoptó de manera definitiva, ofreciendo muy buenos resultados para ambas instancias (Tablas 6.4 y 6.6).

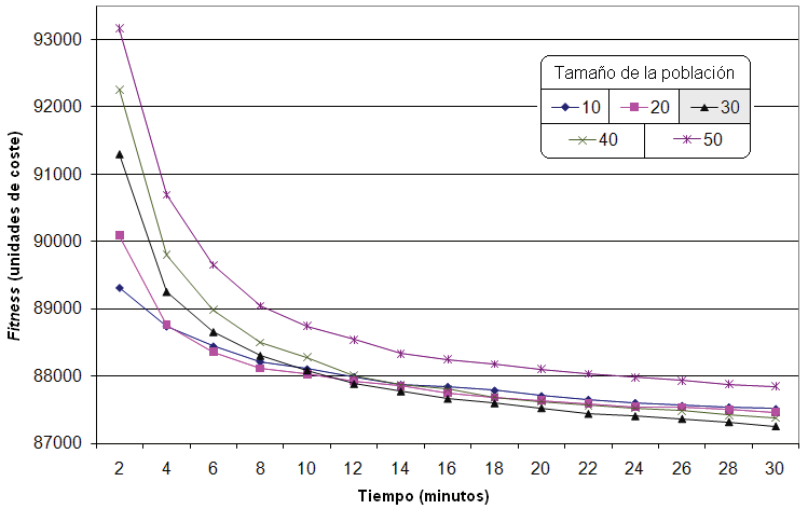

Figura 6.10. Resultados medios de los experimentos llevados a cabo con el tamaño de la población del algoritmo genético aplicado a la instancia Denver

A continuación se experimentó con el tamaño de la población que debía manejar el algoritmo. Dada la experiencia previa, se hicieron pruebas con tamaños de población desde 10 hasta 50 individuos, resultando una población de 30 soluciones como la más adecuada para resolver el problema FAP, como se puede observar en la Figura 6.10. Valores mayores de población ofrecen peores resultados en cualquier franja de tiempo, mientras que valores más pequeños a 30, aunque ofrecen mejores resultados en los primeros minutos de ejecución del algoritmo, debido a que éste evoluciona más deprisa al manejar menos individuos, genera peores planificaciones de frecuencia para ejecuciones más largas, luego el tamaño de la población quedó establecido en 30 soluciones.

El siguiente parámetro que se estudió fue el número de cruces, y por tanto el número de descendientes, y a la postre el número de padres que tendría el algoritmo en cada una de sus generaciones para generar descendencia (para cada descendiente el GA selecciona un par de padres distintos). Dado que el tamaño de la población se estableció a 30 individuos, el número de descendientes quedó limitado como máximo a 15 individuos, sin embargo, tras los experimentos realizados se demostró que aunque este parámetro no resulta crucial en la resolución del problema FAP abordado (no hay grandes mejoras en los resultados con las distintas configuraciones probadas), 6 es el número de descendientes más adecuado, tal y como se ve en la Figura 6.11.

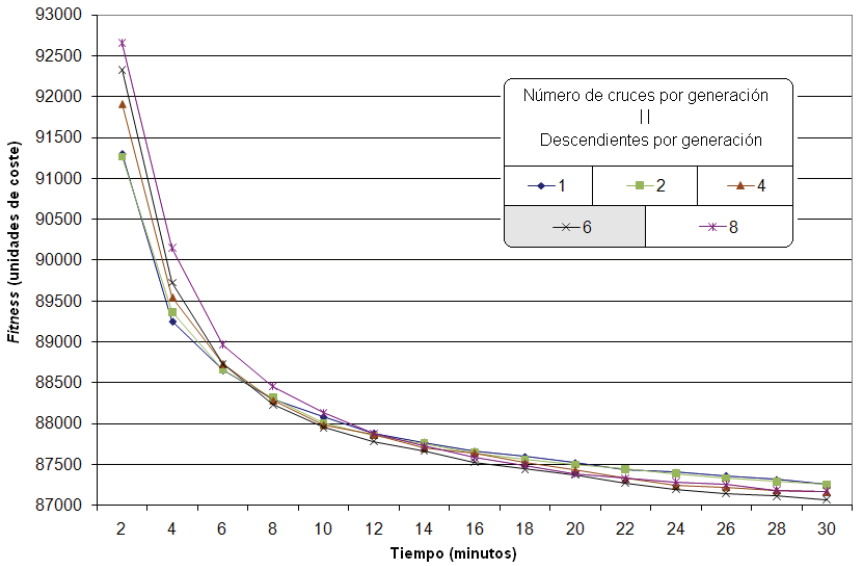

Figura 6.11. Resultados medios de los experimentos llevados a cabo con diferentes números de cruce por generación del GA para la instancia Denver

Menos cruces implican que el algoritmo evoluciona más rápido en los primeros minutos de ejecución, sin embargo también implican una exploración menos exhaustiva, por lo que a la larga se obtienen peores resultados. Por otro lado, añadir más cruces por generación no causa un incremento significativo en la calidad de los resultados, por lo que el valor para este parámetro se estableció en 6. Además, el algoritmo genético incluye de manera implícita una estrategia elitista, ya que se sustituyen las 6 peores soluciones de la población (esta es la mejor estrategia de reemplazo), por lo que los 24 mejores pasan a la siguiente generación.

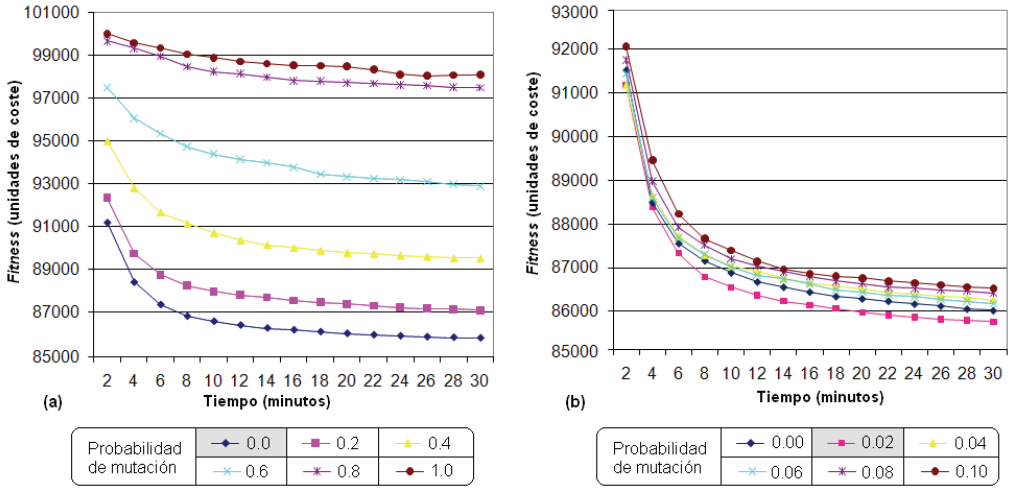

Figura 6.12. Resultados medios de los experimentos llevados a cabo con diferentes probabilidades de mutación del GA para la instancia Denver

Por último, se realizó el estudio paramétrico con la probabilidad de mutación que debía establecerse para el algoritmo. Se hicieron dos tandas de pruebas. La primera con valores que oscilaron desde no incluir ninguna mutación hasta alterar por completo el individuo (Figura 6.12.a). Como se puede observar, la calidad de las planificaciones de frecuencia obtenidas se incrementaba según se reduce la probabilidad de mutación del GA. Por tanto, una primera conclusión que se podía obtener es que aplicar mutación no resulta conveniente para resolver el problema que se aborda, sin embargo, es sabido que la mutación aplicada en momentos clave puede hacer salir del estancamiento la evolución de una metaheurística que se encuentra atrapada en un mínimo local, por lo que se realizó una segunda tanda, con valores más

reducidos de la mutación, para determinar el valor que debía tener este parámetro. El resultados de este experimento, que aparece reflejado en la gráfica de la Figura 6.12.b, muestra que una ligera mutación, con una probabilidad únicamente de 0.02, es el valor más adecuado de configuración para este parámetro para resolver el problema abordado.

En conclusión, tras la realización de los experimentos anteriormente descritos y realizar el pertinente análisis de los mismos, se determinó que la mejor configuración paramétrica del algoritmo genético para resolver el problema de la asignación automática de frecuencias es la siguiente:

- Tamaño de la población = 30 individuos
- Método de selección de padres = torneo binario
- Número de descendientes por generación = 6 soluciones
- Tipo de cruce = cruce uniforme a nivel de TRX
- Operador de mutación = mutación aleatoria, con una probabilidad de mutación de 0.02
- Política de reemplazo = sustitución de los peores individuos

Experimentos y resultados con el algoritmo basado en colonia de abejas

El algoritmo basado en colonia de abejas fue la última de las metaheurísticas basadas en población que se desarrolló. Por tanto, los resultados iniciales que se obtuvieron fueron de mayor calidad que en el resto de metaheurísticas de este tipo, ya que para su desarrollo y ajuste preliminar se tuvo en cuenta la experiencia previa obtenida con el desarrollo de otras técnicas. La metaheurística, como se describió en Algoritmo 5.8, basa su funcionamiento en la exploración de soluciones, fuentes de alimento en este paradigma, que llevan a cabo tres tipos de agentes, abejas, en el espacio de búsqueda del problema. Si una abeja observadora encuentra una solución de calidad, se llama a más abejas, las trabajadoras, para realizar una exploración más exhaustiva en el

entorno de dicha solución, mientras que si es de baja calidad se abandona dicha búsqueda. Además, cada cierto tiempo, las abejas exploradoras buscan aleatoriamente en el espacio de búsqueda para encontrar nuevas fuentes de alimento. Los parámetros de este algoritmo son: el tamaño de la colonia, que se estableció inicialmente a 30 individuos, la proporción de abejas que debía tener la colonia, que de manera estándar se configura aproximadamente con la mitad de la colonia para abejas exploradoras, la otra mitad para trabajadoras y una única abeja exploradora, la frecuencia con la que la abeja exploradora realiza las búsquedas aleatorias, que se configuró inicialmente que debía realizarse en cada iteración del algoritmo, y la probabilidad de mutación, que es el parámetro que utilizan las abejas para realizar la exploración a través de modificaciones en las soluciones (Algoritmo 5.8), y que se configuró de manera inicial a 0.05.

Dado que la mutación es una operación clave en esta metaheurística, lo primero que se hizo fue intentar mejorar la mutación que se venía utilizando hasta ahora. En lugar de utilizar una mutación a nivel de TRX se diseñó una nueva mutación, a nivel sectorial, donde se consideraban un cierto número de TRX dentro del sector a mutar y además se consideraban los sectores que interferían y eran interferidos por la modificación del sector objeto de la operación de mutación. El nuevo diseño, que se pensó en un principio que sería muy prometedor, no mejoró los resultados obtenidos por el operador de mutación utilizado hasta ahora, descrito en el Algoritmo 5.2, por lo que fue éste el que finalmente se utilizó para la colonia de abejas.

A continuación, se estudió el tamaño de la colonia de abejas. Se hicieron experimentos con tamaños entre 10 y 60 soluciones. Según la experiencia adquirida con el análisis de otras metaheurísticas basadas en población, tamaños más pequeños provocan que la metaheurística no realice una exploración adecuada del espacio de búsqueda, mientras que tamaños mayores no permiten que las soluciones evolucionen adecuadamente en un tiempo razonable. La Figura 6.13 muestra un resumen de los resultados obtenidos con los experimentos llevados a cabo para realizar el ajuste del tamaño de la colonia. Como se puede observar, los mejores resultados se obtienen con 30 abejas en la colonia, lo cual es similar al tamaño de la población establecido en los demás algoritmos basados en población (incluso coincide con el valor óptimo que se estableció para el algoritmo genético).

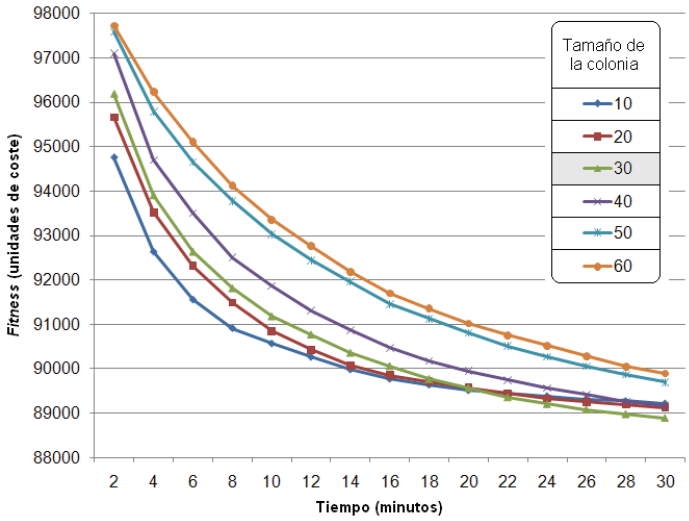

Figura 6.13. Resultados medios de los experimentos llevados a cabo con diferentes tamaños de la colonia para el ABC cuando se aplica a la instancia Denver

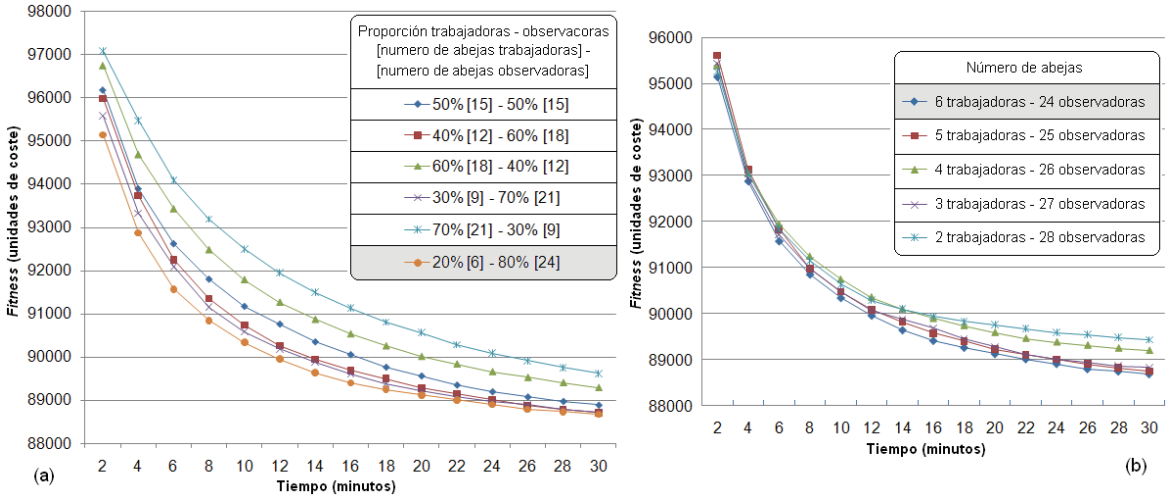

Figura 6.14. Resultados medios de los experimentos llevados a cabo con diferentes proporciones entre abejas exploradoras y trabajadoras

El siguiente parámetro a configurar fue la proporción entre abejas exploradoras y trabajadoras en la colonia. Inicialmente se estableció una proporción del 50% para cada tipo de abejas, y se hicieron pruebas variando ese porcentaje inicialmente un 10% para cada tipo de abejas (Figura 6.14.a), y a continuación realizando un ajuste más fino con el que acotar la proporción óptima (Figura 6.14.b). Como puede observarse en la Figura 6.14, una población

de observadoras grande, que realice una exploración lo más exhaustiva posible del espacio de búsqueda, es la configuración que obtiene soluciones de mayor calidad. Con 6 trabajadoras (del total de 30 que conforman la colonia) que realicen una explotación intensiva en una solución de calidad encontrada por alguna abeja observadora es suficiente, mientras que mantener un número alto de abejas observadoras (24 de un total de 30) garantiza que se realizará una búsqueda más extensiva, y por tanto hay mayores probabilidades de encontrar una solución óptima, por lo que se obtienen mejores resultados.

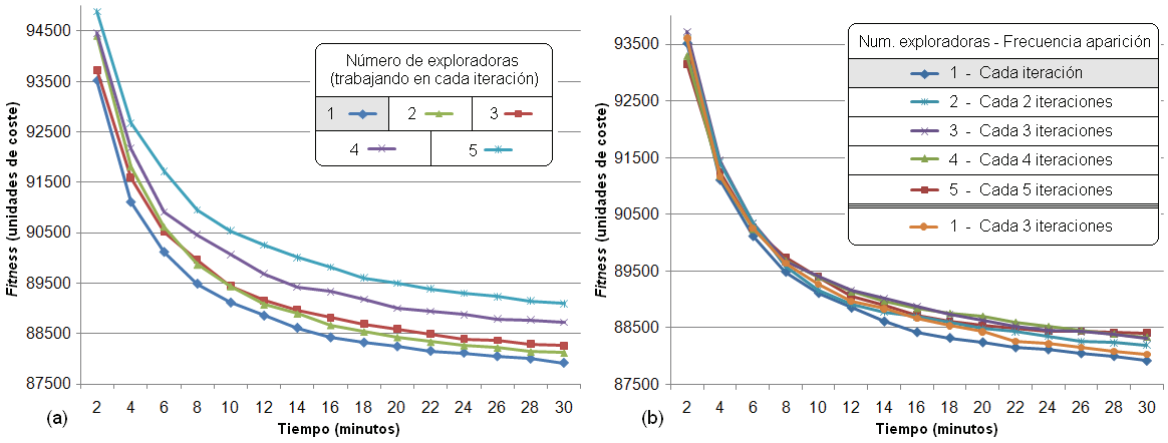

Figura 6.15. Resumen de los resultados medios obtenidos con algunos de los experimentos llevados a cabo con la configuración de las abejas exploradoras

A continuación se hicieron pruebas con el número de abejas exploradoras y la frecuencia con la que estas abejas debían realizar su exploración (generación aleatoria de una nueva solución para la colonia). Según la bibliografía, este número de abejas debe ser bajo, por lo que la primera tanda de experimentos se realizó variando este parámetro de 1 a 5 abejas exploradoras (Figura 6.15.a). A continuación se probaron diferentes frecuencias de aparición con diferentes números de exploradoras, ya que estas abejas no tienen por qué actuar en todas las iteraciones del algoritmo. La Figura 6.15.b muestra un resumen de los resultados que se obtuvieron con varias de las configuraciones comprobadas. No se han incluido todas las pruebas porque resultaron ser bastante irrelevantes, ya que ninguna mejoró los resultados obtenidos con una única abeja exploradora actuando en cada iteración del algoritmo. Esta operación tiene por objetivo hacer que el algoritmo escape de un posible mínimo local, pero su actuación no afecta

a la evolución general del algoritmo, por lo que la configuración que mejores resultado arrojó tiene sentido, ya que la representación de esta abeja debe ser mínima, si bien puede ser necesaria.

Finalmente, se estudió la probabilidad de mutación que debía establecerse en el algoritmo para maximizar la calidad de los resultados. La Figura 6.16 muestra un resumen de los experimentos que se llevaron a cabo con un amplio rango de probabilidades de mutación.

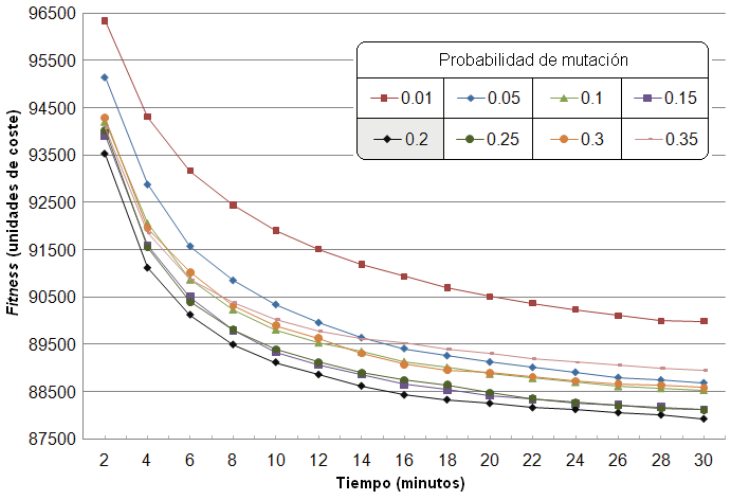

Figura 6.16. Resumen de los resultados medios de los experimentos llevados a cabo con la probabilidad de mutación del ABC en la resolución de la instancia Denver

Como puede observarse, los mejores resultados se obtienen con una probabilidad de mutación de 0.2. Valores mayores de este parámetro empeoran los resultados, al igual que ocurría en el resto de los algoritmos desarrollados, ya que se altera demasiado la solución explorada. Por otro lado, si se produce una alteración demasiado pequeña, no se explora convenientemente el espacio de búsqueda, por lo que los resultados empeoran significativamente. Este parámetro cobra bastante relevancia en el algoritmo ABC, ya que es el operador genético que utiliza la metaheurística para generar nuevas soluciones, por lo que su valor no es tan pequeño como en otras metaheurísticas, como el algoritmo genético, que cuentan con otros operadores genéticos para generar soluciones. Los resultados finales que obtiene el algoritmo basado en la colonia de abejas se muestran en la sección de *Resumen de los resultados*, en las Tablas 6.4 y 6.5 para la instancia Denver y las Tablas 6.6 y 6.7 para la instancia Seattle.

En conclusión, tras la realización de los experimentos anteriormente descritos y realizar el pertinente análisis de los mismos, se determinó que la mejor configuración paramétrica del algoritmo basado en colonia de abejas para resolver el problema de la asignación automática de frecuencias es la siguiente:

- Tamaño de la colonia = 30 soluciones
- Número de abejas trabajadoras = 6 individuos
- Número de abejas observadoras = 24 individuos
- Número de abejas exploradoras = 1 individuo, que reemplaza a la peor solución de la colonia en cada iteración del algoritmo
- Probabilidad de mutación = 0.2

Experimentos y resultados con la búsqueda local iterativa

Debido a su simplicidad y eficiencia (ILS es muy rápido y únicamente posee un parámetro que ajustar), este algoritmo fue la primera de las metaheurísticas basadas en trayectoria que se desarrolló. Como se aprecia en el Algoritmo 5.9, su esquema de funcionamiento es sencillo, ya que únicamente consta de las operaciones de alteración, búsqueda local y actualización durante el tiempo que dure su ejecución. Las operaciones de búsqueda local y actualización carecen de parámetros, mientras que la alteración se lleva a cabo mediante una operación genética de mutación, que como se demostró con los experimentos realizados con las metaheurísticas desarrolladas previamente, debe realizarse a nivel de TRX (Algoritmo 5.2) para que los resultados que se obtienen en el problema que se aborda en esta tesis sean de la mayor calidad posible. La Figura 6.17 muestra un resumen de los experimentos llevados a cabo con la búsqueda local iterativa.

Como se puede observar, se han realizado experimentos con la probabilidad de mutación en un amplio rango, cubriendo valores desde 0.1 hasta 0.9, si bien como en el caso de los algoritmos desarrollados con anterioridad, los mejores resultados se obtienen con valores bajos de este parámetro, ya que la operación de mutación diseñada (Algoritmo 5.2) resulta exhaustiva por sí misma al realizarse a nivel de TRX de la solución. A la vista de los resultados, el mejor

valor medio tras 30 ejecuciones se obtiene con una probabilidad de mutación de 0.2 (al igual que ocurría con el algoritmo basado en colonia de abejas), siendo el siguiente mejor valor 0.1 y reduciéndose desde ese punto la calidad de las planificaciones de frecuencia obtenidas según se incrementa el valor de la probabilidad de mutación (Figura 6.17).

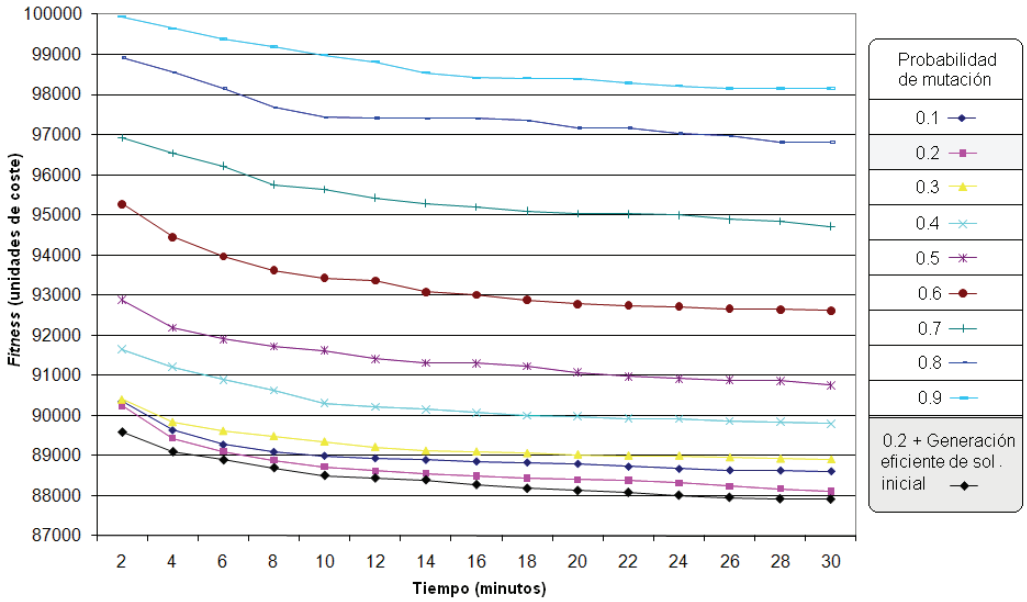

Figura 6.17. Resultados medios de los experimentos llevados a cabo con la búsqueda local iterativa en la resolución de la instancia Denver

Debido a que el algoritmo ILS maneja una única solución que va evolucionando a lo largo de las iteraciones del mismo, se pensó también en que podía ser beneficioso que la solución inicial con la que comenzara a trabajar el algoritmo, en lugar de generarse de manera aleatoria, tal y como se describe en el Algoritmo 5.9 que explica el funcionamiento del mismo en la resolución del problema abordado, se generara de manera eficiente, evitando soluciones que presentaran desde el inicio interferencias con un coste elevado. Por tanto, se desarrolló esta modificación al algoritmo, y tal y como puede observarse en la Figura 6.17, se mejoraron sensiblemente los resultados que se conseguían con la generación aleatoria de la solución inicial, sobre todo en los primeros instantes de ejecución del algoritmo. Sin embargo, esta mejora se ve diluida conforme aumenta el tiempo de ejecución del ILS, resultando la mejora proporcionada muy leve para ejecuciones más largas del algoritmo.

En conclusión, tras realizar los experimentos y analizar los resultados obtenidos con la búsqueda local iterativa, se determinó que la mejor configuración paramétrica para resolver el problema de la asignación automática de frecuencias abordado en esta tesis es la siguiente:

- Probabilidad de mutación = 0.2

Los resultados numéricos exactos conseguidos con esta configuración pueden encontrarse en las Tablas 6.4 y 6.5 para la instancia Denver, y 6.6 y 6.7 para la instancia Seattle.

Experimentos y resultados con la búsqueda de entorno variable

La búsqueda de entorno variable es la metaheurística basada en trayectoria que mejores resultados ha ofrecido en la resolución del problema de optimización abordado en esta tesis. Sin embargo, los resultados que se obtuvieron con la versión estándar del algoritmo no fueron demasiado buenos, siendo superados ampliamente por los del algoritmo ILS. En cambio, la extensión del algoritmo básico llamada SVNS o búsqueda de entorno variable sesgada sí que ofreció muy buenos resultados para el problema FAP. Por esta razón, esta variante de la metaheurística sVNS decidió incluirse en esta tesis, y tal y como se describe en la sección 5.2 (Algoritmo 5.10) consta de dos parámetros: k_{max} y α, que controlan la alteración que se realiza en cada iteración a la solución que está siendo explorada. El primero de los parámetros a configurar fue k_{max}, que si observamos el Algoritmo 5.10, controla la profundidad con la que se realiza la búsqueda en el entorno de la solución con la que se está trabajando. El valor inicial para α quedó establecido en 0.8. La Figura 6.18 muestra un resumen de los resultados obtenidos con los experimentos realizados a este parámetro.

Como se puede observar, el parámetro k_{max} consigue los mejores resultados para la instancia Denver cuando está configurado a su valor máximo, 9, ya que de esta manera se maximiza la búsqueda que se realiza en el entorno de la solución que se está explorando. Un intento de hacer mejorar el algoritmo VNS estándar consistió en repetir cada valor del parámetro k un número

constante de veces, de tal manera que se produjera una búsqueda más exhaustiva en el entorno de cada valor de *k* (durante varias iteraciones), sin embargo, esta modificación no mejoró significativamente los resultados, por lo que no se incluyó en el algoritmo sVNS, que ya posee mecanismos de mejora propios para realizar búsquedas en el entorno de manera más eficiente.

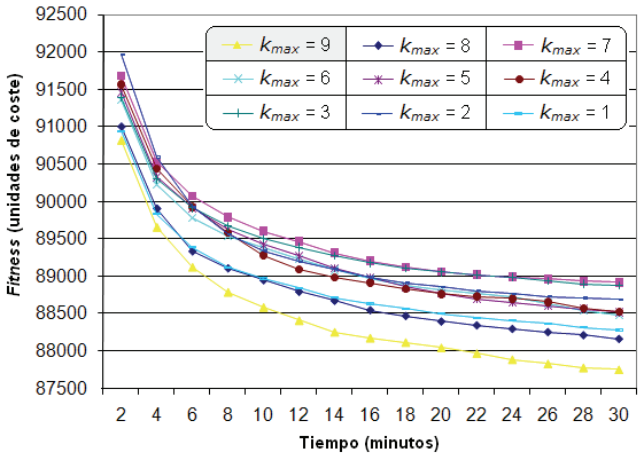

Figura 6.18. Resultados medios de los experimentos llevados a cabo con el parámetro k_{max} del algoritmo sVNS en la resolución de la instancia Denver

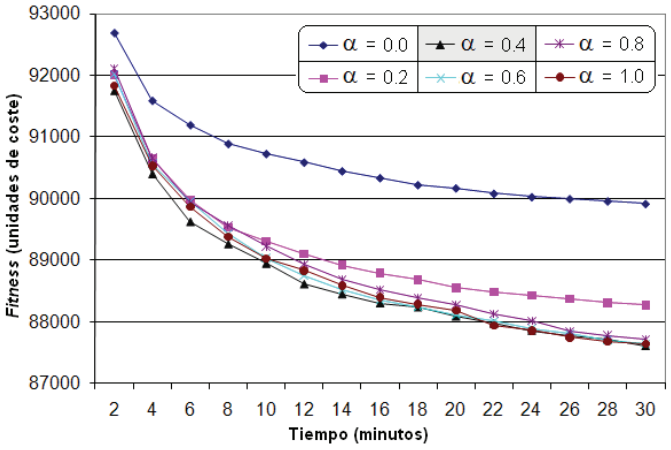

Figura 6.19. Resultados medios de los experimentos llevados a cabo con el parámetro α del algoritmo sVNS en la resolución de la instancia Denver

El otro parámetro a configurar del algoritmo sVNS es el coeficiente α, que modifica la manera de actualizar *k* y por tanto configura la intensidad con la que se explora el entorno de la solución. La Figura 6.19 resume los experimentos

llevados a cabo con este coeficiente, que toma valores de 0 a 1. Como se puede observar, los valores más bajos para este parámetro (0 y 0.2) producen los peores resultados, siendo poco relevante el valor que se tome para valores superiores a 0.4. No obstante, éste es el valor que ofrece los mejores resultados para α en la resolución de la instancia Denver (y aún más en la Seattle) tanto en tiempos de ejecución cortos como en tiempos de ejecución más largos, por lo que es el que finalmente se adoptó.

En conclusión, después de la realización de los experimentos para las instancias abordadas (Sección 6.2) y tras realizar el pertinente análisis de los resultados obtenidos, se determinó que la mejor configuración paramétrica de la búsqueda de entorno variable sesgada para resolver el problema de la asignación automática de frecuencias es la siguiente:

- Valor del parámetro k_{max} = 9
- Valor del parámetro α = 0.4

Experimentos y resultados con el procedimiento de búsqueda avariciosa, aleatoria y adaptativa

El algoritmo GRASP fue desarrollado porque representa una alternativa diferente al resto de metaheurísticas basadas en trayectoria. Pertenece al grupo de algoritmos metaheurísticos que podemos clasificar como métodos constructivos, y su esquema, como puede observarse en Algoritmo 5.11, consta de tres etapas fundamentales: construcción de la solución, mejora de la misma y actualización. La forma de configurar el algoritmo consiste en seleccionar y ajustar el método empleado para construir la solución en cada iteración. Este método puede tener en mayor o menor medida una componente más aleatoria o más avariciosa, dependiendo del tipo de constructor y de los valores de los parámetros configurados, los cuales determinarán la forma de seleccionar las frecuencias que formarán parte de la solución bajo construcción. Además, la probabilidad con la que se puedan seleccionar las frecuencias podrá ser modificada con el sesgo con el que esté configurado el algoritmo. En resumen, los parámetros a

configurar de la metaheurística serán: la versión del GRASP utilizada (tipo 1: construcción basada en cardinalidad, tipo 2: basada en valor, 3: construcción RG y 4: construcción PG) [42, 43], los valores que tendrán los parámetros *k* y α (que indican los elementos que se incluirán en la lista de candidatos a partir de la cual se seleccionan las frecuencias para construir la solución), y la función de sesgo con la que trabaje el algoritmo (se han considerado sesgos aleatorios, lineales y exponenciales).

Los experimentos de ajuste de este algoritmo tienen la particularidad de que fueron realizados en un sistema distribuido grid, mediante scripts que se diseñaron para formar un sistema maestro-esclavo (Figura 4.8) con el que preparar, enviar y recibir los trabajos a la grid. Además, tras dicho ajuste se amplió el modelo maestro-esclavo desarrollado para realizar un pequeño equipo de evolutivos con las distintas versiones del algoritmo ajustadas. Ese diseño no consiguió resultados tan satisfactorios como la hiperheurística paralela descrita en la sección siguiente, pero sirvió para experimentar con un equipo paralelo utilizando computación grid (y también mejoró los resultados de varias de las metaheurísticas incluidas en nuestro estudio, tal y como se discutirá en la sección 6.5 de *Análisis comparativo de resultados*).

Para ajustar el GRASP se llevaron a cabo cuatro bloques de experimentos, que se corresponden con las cuatro versiones del algoritmo utilizadas (tipos 1, 2, 3 y 4), cada uno de los cuales incluyeron las siguientes pruebas: Para la variante basada en cardinalidad (GRASP tipo 1), se realizaron experimentos para ajustar el parámetro *k*, con valores desde 1 hasta 10, y tres posibles sesgos (aleatorio, lineal y exponencial). Para la variante basada en valor (GRASP tipo 2) se realizaron experimentos con valores de α desde 0.01 hasta 0.5 y los tres valores de sesgo. Con la variante RG (GRASP tipo 3) se realizaron experimentos con valores de *k* de 1 a 4 y los tres sesgos. Finalmente, con la variante PG (GRASP tipo 4) el sesgo debe ser aleatorio [43], por lo que únicamente se realizaron pruebas con valores del parámetro *k* de 1 a 4. La Figura 6.20 muestra un resumen de los resultados obtenidos con las mejores configuraciones obtenidas por cada variante del algoritmo.

Como se puede observar, la versión del GRASP que proporciona mejores planificaciones de frecuencia es la variante RG, donde los elementos de la lista de candidatos (Algoritmo 5.11) se seleccionan primero aleatoriamente

(*Random*), y luego de forma avariciosa (*Greedy*), con los valores paramétricos de *k* igual a 1 y el sesgo exponencial. Sin embargo, podemos decir que todas ellas realizan una evolución satisfactoria, donde se reducen significativamente los costes causados por las interferencias de las planificaciones de frecuencia generadas por cada variante. En todo caso, todas las variantes tienen un valor paramétrico para *k* (y α) bajos, lo que significa que los mejores resultados se consiguen cuando el algoritmo no presenta un comportamiento demasiado aleatorio (o en el caso de la variante tipo 4 del GRASP, cuando no hay mucha perturbación en las soluciones).

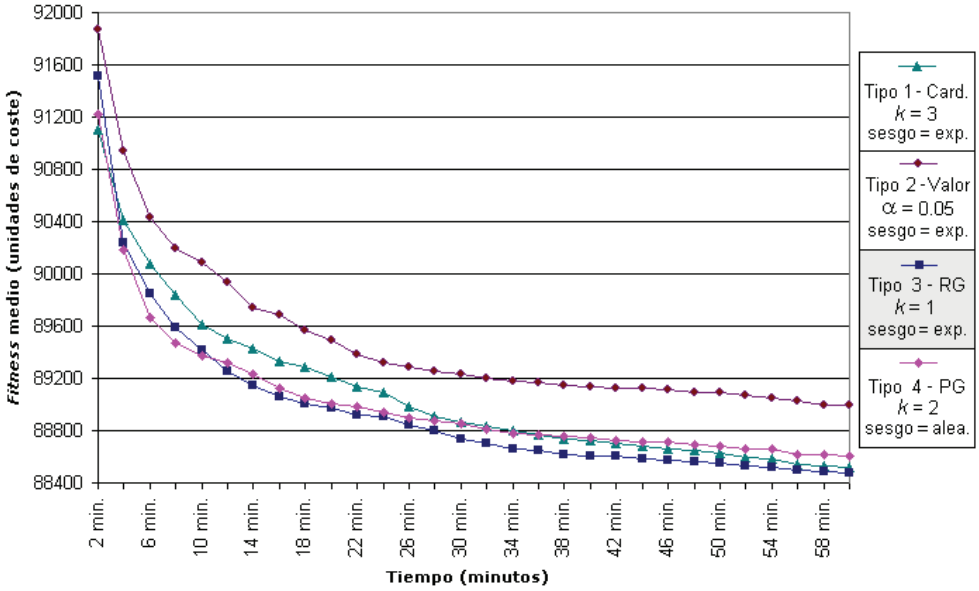

Figura 6.20. Resumen de los resultados obtenidos por las mejores configuraciones de cada tipo de variante del GRASP en la resolución de la instancia Denver

Además, a la vista de los resultados también se pone de manifiesto que el uso de una función de sesgo proporciona una mejora significativa en el comportamiento de cualquiera de las versiones estudiadas. Esto tiene sentido porque con una función de sesgo que no sea aleatoria, las mejores frecuencias tienen mayor probabilidad de ser seleccionadas a la hora de generar una nueva solución. De hecho, las mejores soluciones para los tipos 1, 2 y 3 (el 4 no utiliza sesgo) se obtienen con el sesgo exponencial (Figura 6.20). Por lo tanto, podemos concluir que estrategias no demasiado aleatorias, y bastante avariciosas, son las que obtienen los mejores resultados en la resolución del problema abordado en esta tesis. Finalmente, se pude observar que la variación

tipo 2 del GRASP (Figura 6.20) consigue los resultados más pobres, aún cuando éstos siguen siendo razonablemente buenos.

En conclusión, tras la realización de los experimentos para cuatro variantes del algoritmo GRASP y después de realizar el pertinente análisis de los resultados obtenidos, se determinó que la mejor configuración paramétrica del procedimiento de búsqueda avariciosa, aleatoria y adaptativa para resolver el problema de la asignación automática de frecuencias es la siguiente:

- Tipo de variante = RG
- Valor del parámetro k = 1
- Sesgo = exponencial

Además, como se ha comentado anteriormente, una vez realizado el ajuste de todas las versiones del GRASP, se procedió al desarrollo de un equipo paralelo utilizando una infraestructura grid real [196] a través de un diseño maestro-esclavo distribuido (Figura 4.8). Así, se diseñó un sistema donde se incluyó un representante de cada variante del GRASP con la mejor configuración establecida (los que se incluyen en la Figura 6.20) más una variante extra muy interesante (*tipo* 1, k = 2 y *sesgo* aleatorio) que se seleccionó porque sus resultados eran notablemente buenos utilizando un sesgo aleatorio (lo cual se demostró que no es muy común para el problema abordado). Por tanto, añadimos una quinta variante del GRASP al equipo diseñado con el fin de incluir más diversidad en el sistema paralelo.

El estudio se llevó a cabo haciendo experimentos con diferentes números de nodos en el sistema maestro-esclavo. Se trabajó con 5 (es el mínimo número de nodos porque se incluyeron 5 variantes en el sistema paralelo), 10, 15 y 20 *jobs.* No se pudieron hacer experimentos con un número mayor de *jobs* porque la grid que se utilizó no ofrecía más de 20-25 nodos de trabajo libres como media, lo cual impedía la ejecución de procesos paralelos dependientes del tiempo con un número mayor de requerimientos computacionales.

Es importante explicar también que el sistema paralelo incluía redundancia, ya que cuando se ejecutaban 10, 15 y 20 trabajos, se lanzaron versiones replicadas de los algoritmos en la grid (cada variante del GRASP estaba replicada

las mismas veces que las demás). Esta característica es muy adecuada para los sistemas grid, ya que además de conseguir más resultados en el mismo tiempo, el nodo maestro no tuvo que esperar siempre hasta que todos los trabajos acabaran para evaluar la mejor solución proporcionada por los esclavos.

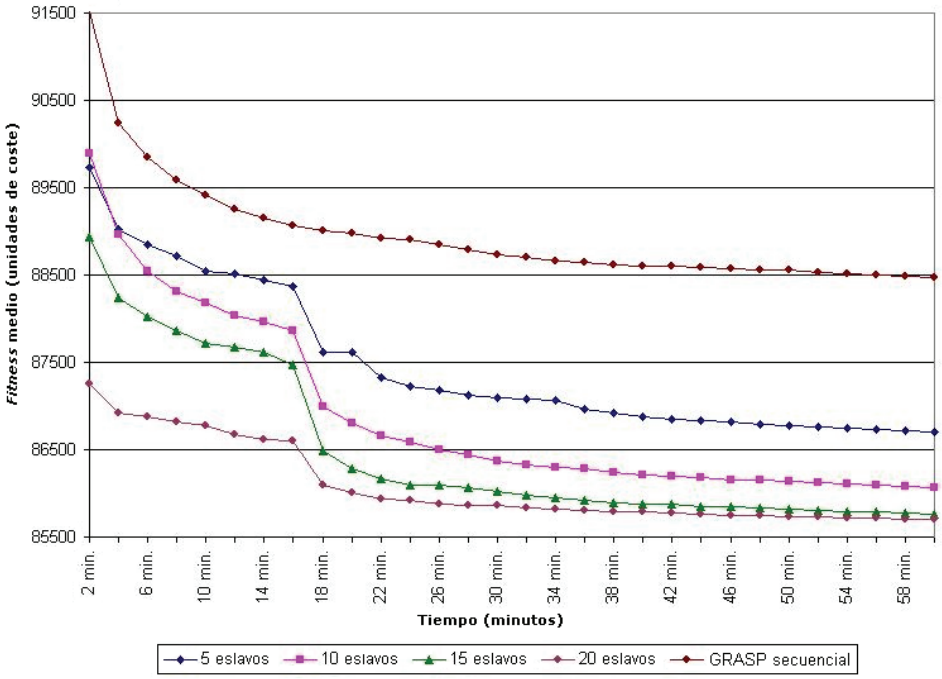

Figura 6.21. Evolución de los resultados obtenidos con diferentes configuraciones del equipo paralelo y la mejor versión secuencial del GRASP

Para el sistema que finalmente se implementó, se configuró que si el 40% de los trabajos acababan, se considera que la iteración había sido realizada con éxito, matando el proceso maestro a los trabajos esclavo que no habían terminado en ese momento (lo cual se debía probablemente a que estaban esperando para ser ejecutados debido a falta de recursos disponibles en la grid o a algún fallo en la infraestructura de la misma). Sin embargo, este hecho casi nunca se produjo, ya que aproximadamente el 90% de los trabajos lanzados terminaban correctamente sus ejecuciones, devolviendo a tiempo los resultados correspondientes. La Figura 6.21 muestra un resumen de los resultados obtenidos con diferentes configuraciones del sistema paralelo desarrollado cuando se abordó la resolución de la instancia Denver del problema (Sección 6.2).

Como se puede observar, todas las versiones paralelas mejoran a la mejor variante secuencial del algoritmo, obteniéndose mejores resultados cuanto más nodos participan en la ejecución del algoritmo. Además, se aprecia una gran mejora a los 16 minutos de ejecución. Este hecho se explica porque el sistema es sincronizado cada 16 minutos (se sincroniza 3 veces en una hora de ejecución), y en el minuto 16 se produce la primera de estas sincronizaciones, donde cada uno de los esclavos envía al nodo maestro la mejor solución encontrada hasta ese momento, y éste reenvía dicha solución a cada nodo para que continúen con esta solución su ejecución. Por tanto, la primera sincronización produce que todas las versiones mejoren sustancialmente los resultados en este punto. El resto de sincronizaciones no representan saltos tan marcados en la gráfica de la Figura 6.21 porque las mejoras son más suaves. Para ampliar el análisis de los resultados conseguidos con esta versión paralela del GRASP consúltese la referencia [56], donde se encuentra la publicación que se consiguió con el trabajo llevado a cabo con este sistema.

Tabla 6.3. Resultados para tres límites temporales conseguidos por las mejores versiones secuencial y paralela del GRASP en la resolución de la instancia Denver

	120 segundos		600 segundos		1800 segundos	
	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$
GRASP secuencial	88334.0	90328.3±1121.7	86837.5	89073.5±992.5	86216.7	88599.3±776.8
GridGRASP	85313.0	87256.9±2309.2	85313.0	86772.1±1701.0	85259.4	85855.3±686.9

La Tabla 6.3 muestra los resultados finales obtenidos tanto por la versión secuencial como por la versión paralela del GRASP. La versión paralela del GRASP mejora claramente los resultados obtenidos por la versión secuencial, si bien no representa una mejora significativa para los resultados obtenidos con otras metaheurísticas estudiadas en esta tesis, como son la búsqueda dispersa o el algoritmo genético. Sin embargo, muchas de las ideas aplicadas en el desarrollo de la versión paralela del GRASP utilizando grid fueron aplicadas a la hiperheurística paralela que se desarrolló con posterioridad (ver capítulo 5.3), cuyos resultados se discutirán en la sección 6.4.

Resumen de los resultados obtenidos por las metaheurísticas

Como se ha explicado en las secciones anteriores, todas las metaheurísticas incluidas en nuestro estudio han sido ajustadas cuidadosamente para que obtuvieran resultados de la máxima calidad posible a la hora de resolver el problema abordado en esta tesis. Las Tablas 6.4 y 6.5 muestran un resumen de los resultados obtenidos en cuatro límites temporales para la instancia Denver. De esta forma, se muestran los resultados medios junto con las desviaciones estándar (Tabla 6.4), así como los mejores costes conseguidos (Tabla 6.5) en la resolución de dicha instancia. Por otro lado, las Tablas 6.6 y 6.7 muestran los mismos resultados pero para la instancia Seattle.

Tabla 6.4. Resumen de los resultados medios obtenidos en 30 ejecuciones independientes y 4 límites temporales para las metaheurísticas desarrolladas en esta tesis cuando se aplican a la resolución de la instancia Denver

Denver	120 segundos $\bar{x}\pm\sigma$	600 segundos $\bar{x}\pm\sigma$	1800 segundos $\bar{x}\pm\sigma$	3600 segundos $\bar{x}\pm\sigma$
SS	88692.7±1124.9	86843.8±950.5	85767.6±686.3	85455.9±432.2
GA	90586.8±1083.2	86680.1±974.3	85994.9±680.5	85674.7±654.4
ABC	92383.8±1005.3	88428.7±815.3	87432.5±789.3	87098.6±544.1
PBIL	98659.0±1215.2	90435.8±1007.2	88931.6±968.5	88688.6±742.0
ILS	89993.8±1319.4	88592.4±1169.3	88167.8±995.2	87891.1±844.7
VNS	91380.0±1034.4	88896.2±821.8	87665.0±728.9	86957.3±502.1
GRASP	90328.3±1296.7	89073.5±1002.5	88599.3±976.8	88354.7±807.8

Tabla 6.5. Resumen de los mejores resultados obtenidos en 30 ejecuciones independientes y 4 límites temporales para las metaheurísticas desarrolladas en esta tesis cuando se aplican a la resolución de la instancia Denver

Denver	120 segundos	600 segundos	1800 segundos	3600 segundos
SS	86597.3	85800.9	84942.5	84205.4
GA	88529.7	85253.6	84994.9	84543.7
ABC	90361.0	86463.5	86045.8	85697.7
PBIL	97038.8	88734.2	87366.0	87212.1
ILS	87589.4	86642.2	86167.2	86167.2
VNS	89038.3	87305.5	86643.8	85755.0
GRASP	88334.0	86837.5	86216.7	86216.7

Como se puede observar a la vista de las cuatro tablas, todas las metaheurísticas consiguen planificaciones de frecuencia de muy alta calidad, con desviaciones estándar que además no son demasiado altas, lo cual demuestra que los resultados conseguidos son bastante fiables, como además se corroborará en el Apéndice I de *Análisis estadístico de los resultados*.

Tabla 6.6. Resumen de los resultados medios obtenidos en 30 ejecuciones independientes y 4 límites temporales para las metaheurísticas desarrolladas en esta tesis cuando se aplican a la resolución de la instancia Seattle

Seattle	120 segundos $\bar{x}\pm\sigma$	600 segundos $\bar{x}\pm\sigma$	1800 segundos $\bar{x}\pm\sigma$	3600 segundos $\bar{x}\pm\sigma$
SS	1891.0±194.9	1508.1±160.5	1282.4±146.3	1154.0±140.2
GA	1276.3±177.5	826.3±119.2	765.2±87.9	745.5±72.1
ABC	1889.5±143.1	1491.2±109.8	1319.4±98.5	1236.7±85.3
PBIL	2420.0±188.3	1698.7±143.5	1572.6±137.3	1521.0±130.4
ILS	1703.6±174.3	1439.8±133.0	1319.8±125.4	1251.5±118.3
VNS	1568.7±229.3	1123.1±179.3	939.2±143.0	825.1±122.6
GRASP	2108.1±172.1	1928.6±165.7	1857.2±134.2	1799.2±110.7

Tabla 6.7. Resumen de los mejores resultados obtenidos en 30 ejecuciones independientes y 4 límites temporales para las metaheurísticas desarrolladas en esta tesis cuando se aplican a la resolución de la instancia Seattle

Seattle	120 segundos	600 segundos	1800 segundos	3600 segundos
SS	1543.2	1245.6	1051.8	913.1
GA	950.9	656.8	613.7	609.6
ABC	1611.3	1277.5	1128.5	1084.5
PBIL	2139.1	1401.5	1342.0	1222.2
ILS	1406.6	1215.4	1023.7	908.8
VNS	1075.5	755.9	666.6	630.9
GRASP	1815.3	1754.9	1595.8	1595.8

En caso de la instancia Denver, los mejores resultados son conseguidos por las metaheurísticas basadas en población SS y GA, y por la basada en trayectoria VNS, mientras que para la instancia Seattle, las planificaciones de frecuencia de mayor calidad son conseguidas por el GA y el algoritmo VNS. Por otro lado, los resultados de menor calidad para ambas instancias son obtenidos por los algoritmos PBIL y GRASP, que son metaheurísticas particularmente

diferentes al resto, ya que la primera está basada en un modelo probabilístico que utiliza aprendizaje competitivo para hacer evolucionar a un vector de probabilidad que representa la tendencia global de la población, mientras que el GRASP es un método constructivo que hace evolucionar una solución que se construye en cada iteración mediante un método en parte avaricioso y en parte aleatorio. El resto de metaheurísticas hacen evolucionar tanto a su población (caso de trabajar con un conjunto de soluciones), como a la única solución con la que trabajan (caso de las metaheurísticas basadas en trayectoria), mediante operaciones genéticas tradicionales, como son el cruce uniforme, explicado en el Algoritmo 5.3, y la mutación aleatoria, explicada en el algoritmo 5.2.

Además, todos los algoritmos presentan patrones de configuración comunes para varios de sus parámetros. Por ejemplo, las metaheurísticas basadas en población funcionan mejor con poblaciones no demasiado grandes, de entre 10 y 30 individuos aproximadamente, ya que SS trabaja con un conjunto de referencia de 9 soluciones, 30 son los individuos configurados para el GA y el algoritmo ABC, mientras que PBIL tiene una población de 20 individuos. Por otro lado, la operación de cruce uniforme a nivel de TRX (Algoritmo 5.3) proporciona muy buenos resultados, ya que las aproximaciones que lo incluyen, que son SS y GA proporcionan prácticamente los resultados de mayor calidad de nuestro estudio. También se ha demostrado que la mutación es un parámetro que debe utilizarse con bastante prudencia, ya que cuando la evolución de las soluciones dependen de algún otro operador genético, este parámetro debe tener un valor muy bajo, como son los casos de PBIL o GA, donde la probabilidad de mutación tiene un valor de 0.02, o incluso en SS, donde la mutación llega a desaparecer. Por otro lado, cuando la evolución de las soluciones se debe únicamente a alteraciones debidas a operadores de mutación, este valor tiene que ser mayor, como se demuestra para el caso del algoritmo ABC o de ILS, que presentan probabilidades de mutación de 0.2.

Por otro lado, todos los algoritmos presentan otras características comunes que proporcionan un mejor funcionamiento de todos ellos, como son la aplicación de estrategias elitistas en todas las metaheurísticas. De esta forma, siempre se preservan las mejores soluciones conseguidas en cada iteración, siendo la política de reemplazo de todas las aproximaciones la sustitución de las soluciones de peor calidad. Además, en todos los algoritmos predomina su

componente avariciosa sobre su componente aleatoria, ya que todos comparten una búsqueda local muy eficiente adaptada al problema que se aborda (Algoritmo 5.1), todos presentan una probabilidad de mutación bastante baja, y todos evolucionan utilizando operadores configurados de manera avariciosa.

6.4. Experimentos y resultados con la hiper-heurística paralela

En esta sección se describen los experimentos y se analizan los resultados obtenido con la hiperheurística paralela desarrollada. El objetivo de las distintas pruebas fue el de configurar el sistema paralelo para que éste fuera capaz de obtener resultados de la más alta calidad en la resolución del problema FAP planteado en esta tesis. Como se mostrará en esta sección, las planificaciones de frecuencia conseguidas por la hiperheurística diseñada en esta tesis representan, hasta donde nosotros sabemos, los mejores resultados que se han publicado hasta la fecha en la resolución de las instancias reales del FAP con las que se trabaja en esta tesis.

Como se explicó en el capítulo anterior, la hiperheurística paralela (PHH – *parallel hyper-heuristic*) está basada en un modelo maestro-esclavo donde el proceso maestro del sistema distribuye instancias de las metaheurísticas que se han descrito en las secciones previas (PBIL, SS, GA, ABC, ILS, VNS y GRASP) entre los procesadores disponibles del sistema (Figura 5.2). Cada cierto periodo de tiempo, el cuál es un parámetro a configurar del sistema, el proceso maestro (Algoritmo 5.12) recupera la mejor solución encontrada por cada metaheurística. El sistema utilizará esta información en las sincronizaciones que se llevan a cabo entre el maestro y los procesos esclavos, tal y como se describió en la Sección 5.3. El proceso de sincronización es muy importante, ya que por un lado, se retroalimenta a cada metaheurística con la mejor solución encontrada hasta ese momento por el sistema global, para que cada algoritmo continúe su ejecución desde ese punto con dicha solución, mientras que por otro lado, también se determina cuántos núcleos de procesamiento del sistema ejecutarán cada metaheurística desde una sincronización a la siguiente. Por tanto, la frecuencia

con la que la PHH realice las sincronizaciones es un parámetro importante en el sistema que redundará en la calidad de las soluciones que se obtengan. En este sentido, si el número de sincronizaciones es demasiado alto, el sistema gastará demasiado tiempo en esta tarea, dejando a las metaheurísticas sin tiempo suficiente para hacer evolucionar las soluciones. Por el contrario, si se realizan menos sincronizaciones de las debidas, las metaheurísticas trabajarán demasiado tiempo de manera aislada, como islas independientes, por lo que los resultados tampoco serán de tanta calidad como deberían. Por tanto, para llevar a cabo el ajuste de las sincronizaciones, se ha llevado a cabo una serie de experimentos utilizando las instancias descritas en la Sección 6.2 con el objetivo de determinar la frecuencia óptima de sincronizaciones que debe realizar la PHH para obtener planificaciones de frecuencia de la mayor calidad posible para las redes de telefonía abordadas. Las Figuras 6.22 y 6.23 resumen los resultados obtenidos con dichos experimentos. Se han realizado pruebas sincronizando el sistema cada minuto, cada 2 minutos, cada 5, 10 y 15 minutos. Teniendo en cuenta que la duración total del experimento es de 30 minutos (Sección 6.3) significa que se han hecho pruebas realizando 29, 14, 5, 2 y 1 sincronizaciones respectivamente.

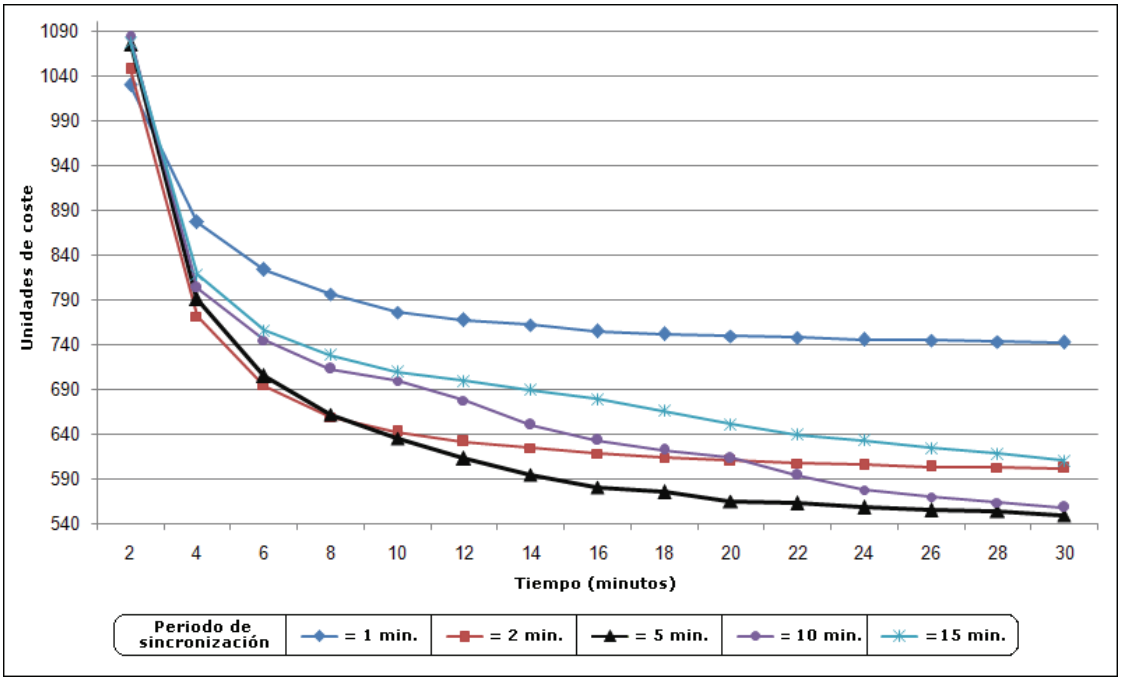

Figura 6.22. Evolución en los resultados obtenidos para la instancia Seattle utilizando diferentes sincronizaciones

Como se puede observar tanto en la Figura 6.22 como en la 6.23, los mejores resultados se obtienen cuando la PHH sincroniza las metaheurísticas cada 5 minutos. Este hecho se observa claramente en el caso de la instancia Seattle (Figura 6.22), ya que los resultados conseguidos con esta configuración tienen un coste menor que los resultados sincronizando cada 2 y cada 10 minutos. Sincronizar cada 10 minutos nunca es más provechoso que cada 5, mientras que un periodo de sincronización de 2 minutos sólo es ligeramente mejor durante los 8 primeros minutos de ejecución (donde el sistema sólo se ha sincronizado una vez con la configuración de realizar sincronizaciones cada 5 minutos).

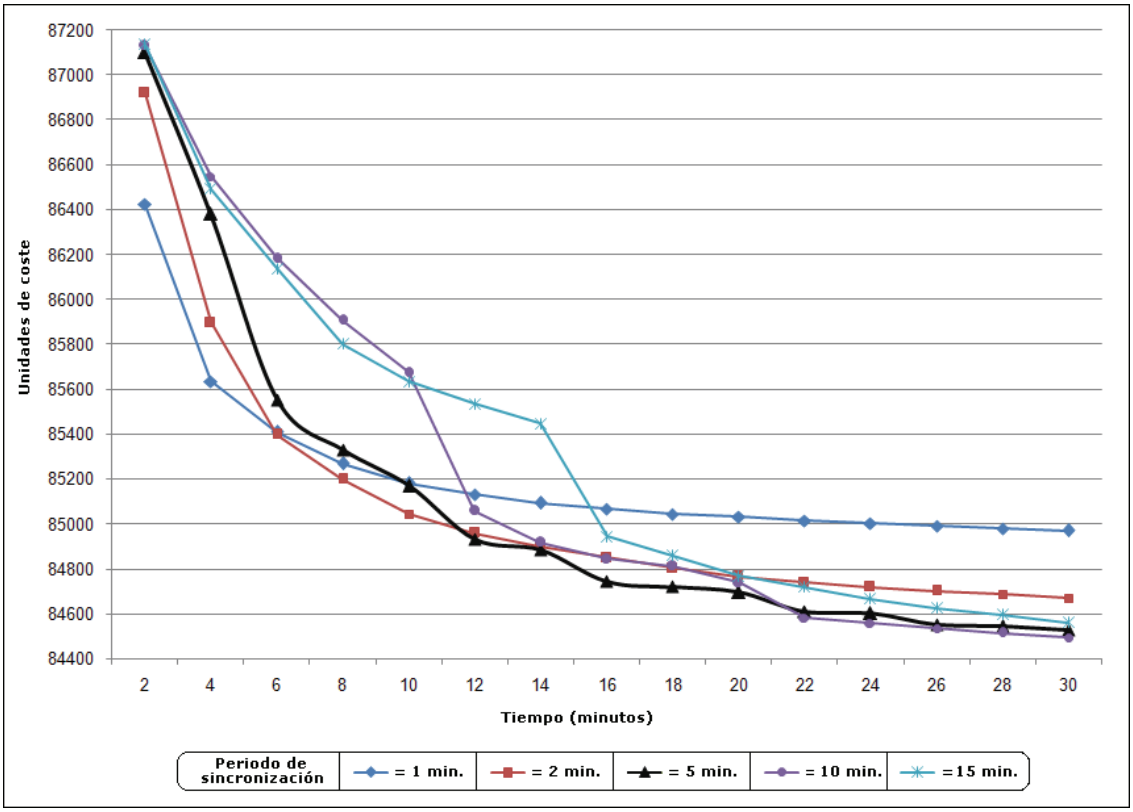

Figura 6.23. Evolución en los resultados obtenidos para la instancia Denver utilizando diferentes sincronizaciones

En la Tabla 6.8, la fila correspondiente a la instancia Seattle muestra el mejor caso, la media y la desviación estándar de 30 ejecuciones independientes con la mejor configuración encontrada para tres límites temporales (2, 10 y 30 minutos). Si se configura al sistema con un número mayor de sincronizaciones

los resultados son también muy buenos en los primeros minutos de ejecución del algoritmo (Figura 6.22), sin embargo, empeoran mucho tras estos primeros instantes, ya que se invierte demasiado tiempo en la operación de sincronización, por lo que las metaheurísticas no tienen tiempo suficiente para hacer que las soluciones evolucionen adecuadamente. Por tanto, queda claro que sincronizar cada 5 minutos es la mejor opción para obtener las planificaciones de frecuencia de la mayor calidad a la hora de resolver la instancia Seattle.

Por otro lado, si se analizan los datos para la instancia Denver (Figura 6.23), se puede observar que los mejores resultados a corto plazo son alcanzados por configuraciones que realizan sincronizaciones cada muy poco tiempo (1 o 2 minutos), sin embargo, a partir de los 10 minutos, las mejores planificaciones son obtenidas si el sistema se sincroniza cada 5, 10 o incluso 15 minutos. De hecho, si se considera la mejor configuración de Denver para los 30 minutos completos de ejecución, que es sincronizar cada 10 minutos, se obtiene un resultado medio de 84484.53 unidades de coste. No obstante, si se establece sincronizar cada 5 minutos, el sistema consigue mejores resultados medios en muchos más periodos de tiempo, tal y como se puede observar en la Figura 6.23, donde la línea de los 5 minutos representa la mejor configuración considerando de manera global los 30 minutos de ejecución. El resultado alcanzado con esta configuración puede observarse en la fila correspondiente a la instancia Denver de la Tabla 6.8.

Tabla 6.8. Resultados empíricos de la hiperheurística paralela desarrollada realizando sincronizaciones cada 5 minutos. Se indican tres límites temporales diferentes para las dos instancias abordadas, Seattle y Denver. Se muestra la mejor solución, la media ($\bar{x}$) y las desviaciones estándar (σ) de 30 ejecuciones independientes

	120 segundos		600 segundos		1800 segundos	
	Mejor	$\bar{x}$±σ	Mejor	$\bar{x}$±σ	Mejor	$\bar{x}$±σ
Seattle	925.44	1075.01±59.86	546.76	635.61±43.01	456.64	550.07±46.50
Denver	86340.26	87100.20±381.16	84246.64	85167.94±382.86	83815.78	84527.43±404.72

Tras el análisis anteriormente expuesto se puede concluir que, para las dos instancias abordadas, las mejores planificaciones de frecuencia se obtienen cuando la hiperheurística paralela sincroniza las metaheurísticas en las que se

apoya cada 5 minutos. Un número mayor de sincronizaciones hace que se dedique demasiado tiempo a esta tarea, no dejando tiempo suficiente a las metaheurísticas para hacer evolucionar las soluciones, y por tanto obteniendo resultados pobres en ejecuciones superiores a 10 minutos. Por el contrario, con menos sincronizaciones, las metaheurísticas trabajan como islas independientes demasiado tiempo, consiguiendo peores planificaciones de frecuencia debido a la falta de control ejercida por la hiperheurística. Los resultados finales para ambas instancias pueden observarse en la Tabla 6.8.

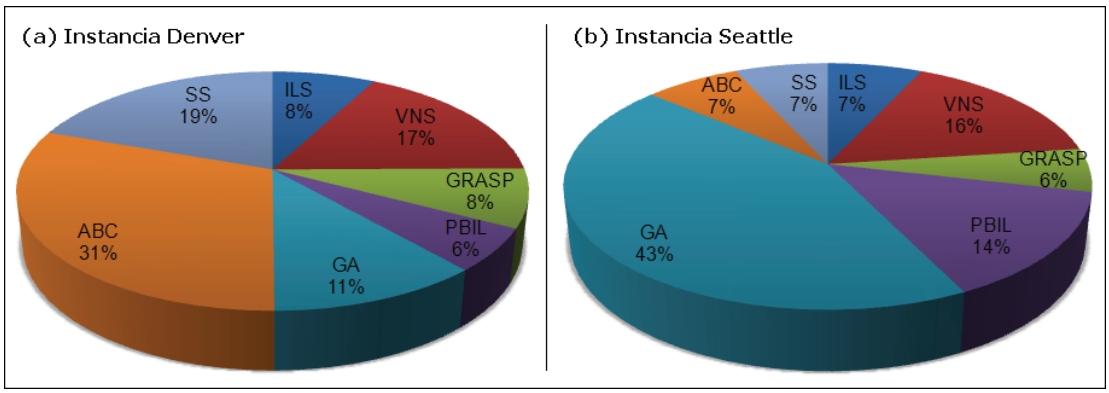

Figura 6.24. Aportaciones de las distintas metaheurísticas al sistema paralelo para las instancias Denver (a) y Seattle (b)

Además, se ha analizado la contribución que cada metaheurística realiza a la evolución global del sistema. Sólo se mostrarán los resultados de la mejor configuración de sincronizaciones (sincronizaciones cada 5 minutos). La primera conclusión que se puede extraer de este análisis es que todas las metaheurísticas colaboran en la evolución global del sistema, ya que todas ellas aportan soluciones de calidad que hacen a su vez evolucionar las soluciones de todos los algoritmos. Sin embargo, como se observa en la Figura 6.24, las mejores planificaciones de frecuencia son aportadas con una proporción bastante grande por las metaheurísticas basadas en población. Este hecho se comprueba analizando las contribuciones para ambas instancias abordadas. En caso de la instancia Denver, el algoritmo ABC es el método que realiza las mejores contribuciones al sistema, y la suma de las contribuciones del ABC, SS y GA suman aproximadamente el 60% de los mejores individuos generados por el sistema. Este hecho se explica debido a que las metaheurísticas basadas en

población son las más estocásticas, y realizar sincronizaciones frecuentemente, lo cual les proporciona a intervalos constantes contribuciones de calidad, hace que se aporte un componente avaricioso que hace que mejoren en gran medida los resultados que ofrecen. Por otro lado, las metaheurísticas basadas en trayectoria son avariciosas *per se* por lo que las sincronizaciones tienen menos impacto en su funcionamiento.

La instancia Seattle presenta un comportamiento similar, pero en este caso es el algoritmo genético (GA) el que proporciona más del 40% de las mejores soluciones en las sincronizaciones del sistema, mientras que el resto de metaheurísticas comparten las contribuciones que realizan al sistema. Destaca en este caso las buenas contribuciones aportadas por PBIL y VNS, que ponen de manifiesto que incluso las metaheurísticas basadas en trayectoria, que han obtenido resultados más pobres en este estudio, contribuyen en algún momento en la evolución efectiva del sistema global. De hecho, estos algoritmos funcionan mejor que los basados en población durante los primeros minutos de ejecución, ya que evolucionan más rápidamente, por lo que su contribución es relevante a la hora de obtener planificaciones de frecuencia de alta calidad al final del proceso.

6.5. Análisis comparativo de resultados

En esta sección se analizan y comparan los resultados obtenidos por los algoritmos desarrollados en esta tesis. En primer lugar, se cotejarán los resultados de las distintas técnicas desarrolladas en esta tesis, examinando el funcionamiento y evolución de todas ellas, y poniendo de manifiesto la mejora que se ha alcanzado con la combinación de todas las metaheurísticas en la hiperheurística paralela desarrollada. A continuación se compararán los resultados obtenidos en nuestra investigación con otros resultados relevantes publicados en la bibliografía, lo cual pondrá de manifiesto la calidad de éstos y la validez de las propuestas realizadas en esta tesis.

En la Figura 6.25 se muestra una gráfica comparativa que expresa los resultados obtenidos por cada una de las metaheurísticas desarrolladas y de la hiperheurística paralela final para la instancia del problema Seattle.

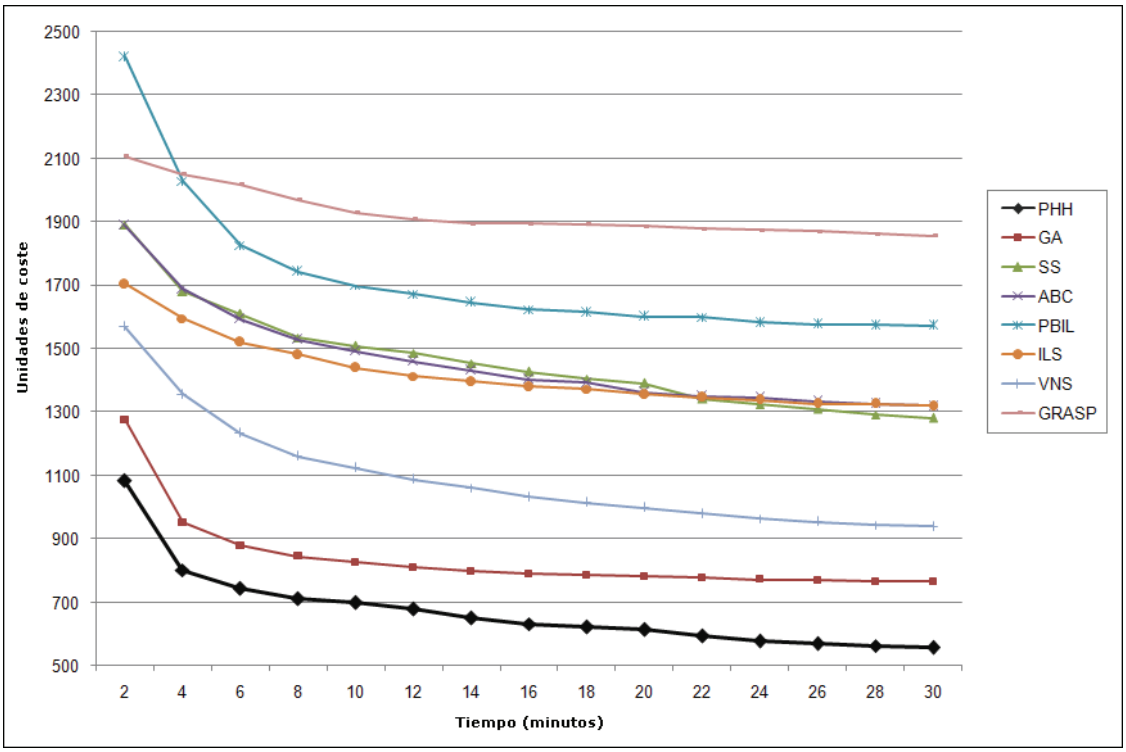

Figura 6.25. Comparativa de resultados de cada una de las estrategias desarrolladas en esta tesis para la instancia de Seattle

Como se puede observar, todos los algoritmos presentan una evolución satisfactoria, destacando la mejora aportada por el sistema paralelo desarrollado en cualquier periodo de tiempo (desde los 2 hasta los 30 minutos), con una mejora constante en los valores de *fitness* de las soluciones generadas. Además, también se observa cómo la metaheurística que contribuye más en el sistema paralelo (GA, Figura 6.24.b) es también el algoritmo que mejores resultados genera de manera individual (Figura 6.25), si bien esto no puede aplicarse como normal general. Prueba de ello es, por ejemplo, el algoritmo PBIL, que obtiene resultados individuales bastante pobres (Figura 6.25), pero que en el sistema paralelo es el tercer algoritmo que mejores soluciones aporta (Figura 6.24.b). Por tanto, todos los algoritmos contribuyen activamente en algún momento de la evolución del sistema paralelo, aunque algunos presentan resultados individuales de mayor calidad que otros.

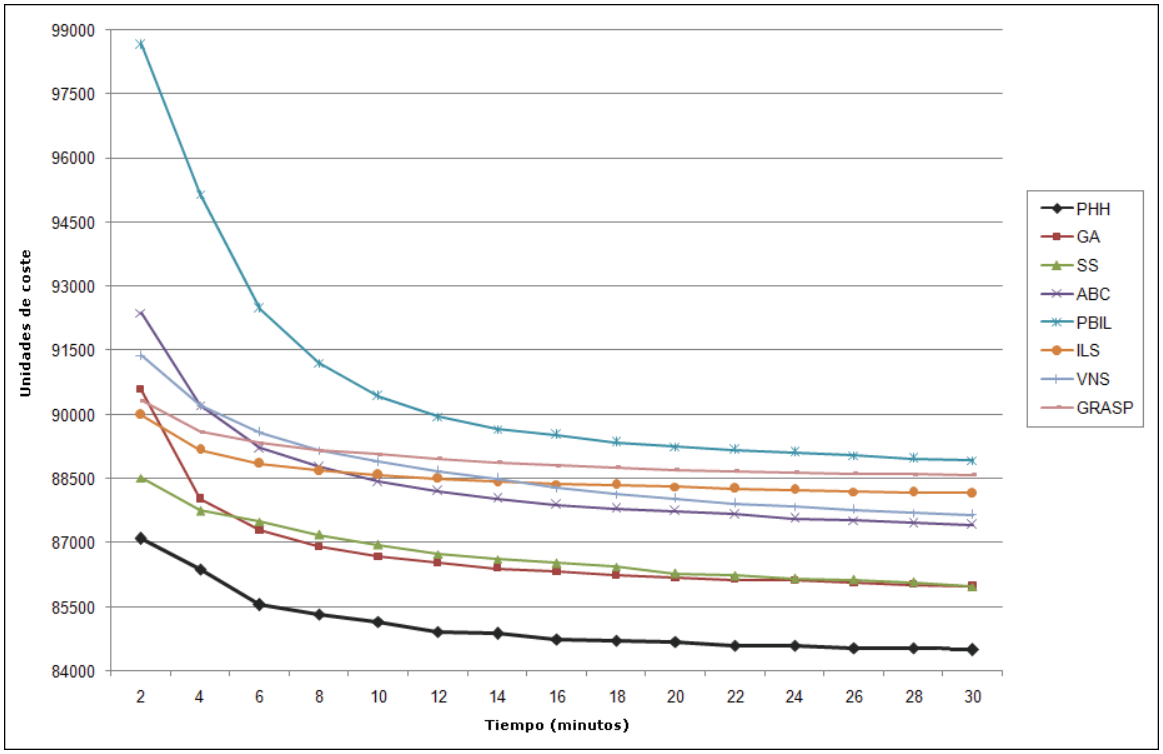

Figura 6.26. Comparativa de resultados de cada una de las estrategias desarrolladas en esta tesis para la instancia de Denver

Por otra parte, la Figura 6.26 muestra los resultados medios de las técnicas desarrolladas aplicadas a la instancia de Denver (Sección 6.2). La mejora ofrecida en este caso por la hiperheurística paralela respecto a las metaheurísticas secuenciales es aún mayor que en el caso de la instancia Seattle. Además, los resultados medios obtenidos por las metaheurísticas son bastante similares. Destacan los algoritmos SS y GA, que mejoran claramente los resultados del resto (Figura 6.26). Sin embargo, este hecho se contrapone con el estudio de las aportaciones a la hiperheurística (Figura 6.24.b), ya que el algoritmo que más soluciones de alta calidad aporta en las sincronizaciones del sistema paralelo para la instancia Denver es el algoritmo ABC (Figura 6.24.b), que genera soluciones de peor calidad media que los algoritmos SS y GA de manera individual (Figura 6.26).

Sin embargo, no es justo en términos computacionales realizar una comparación de la hiperheurística paralela con las metaheurísticas secuenciales, por lo que se realizaron también pruebas con un sistema paralelo en el que en lugar de tener siete metaheurísticas que son gestionadas por una

hiperheurística, se tiene un equipo homogéneo con tantas instancias de la misma metaheurística como instancias heterogéneas poseía la hiperheurística. De esta forma, se comparó un equipo heterogéneo de metaheurísticas (hiperheurística paralela) con un equipo homogéneo que poseía la misma capacidad de cómputo (mismo número de cores y de metaheurísticas trabajando a la vez). La Figura 6.27 muestra el resultado de este estudio comparativo para la instancia Seattle, y como se puede observar, la PHH mejora significativamente los resultados de cualquier equipo homogéneo para cualquier rodaja de tiempo estudiada (entre los 2 y los 30 minutos).

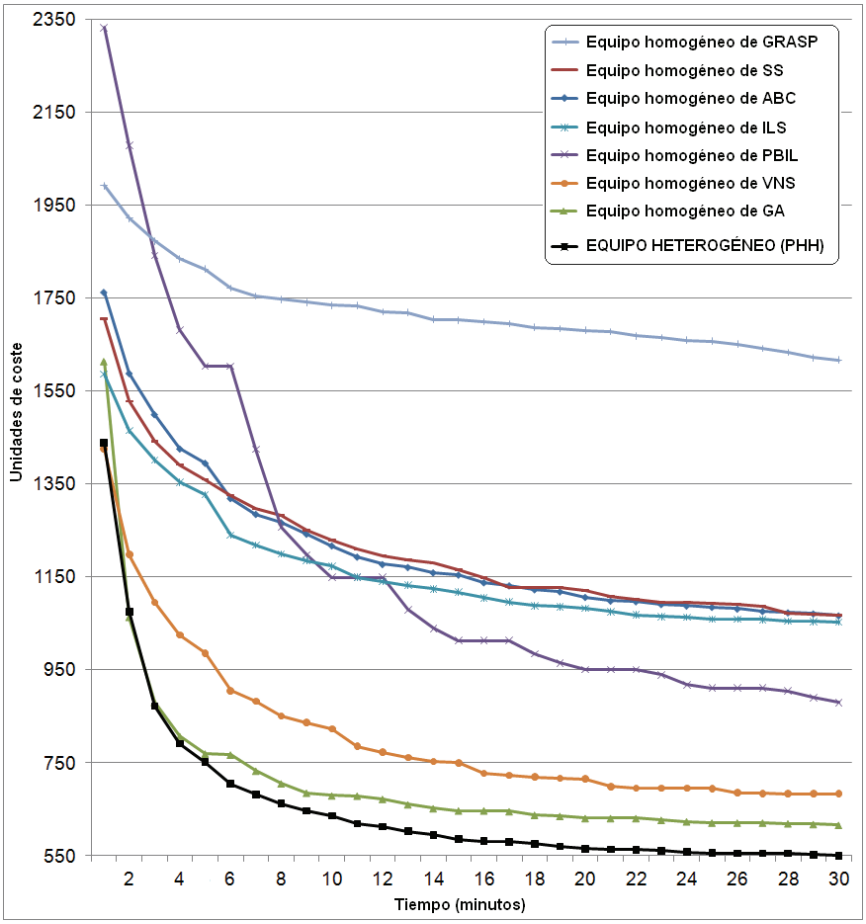

Figura 6.27. Comparativa de resultados entre la hiperheurística paralela (PHH, equipo heterogéneo) y siete equipos homogéneos formados por la misma metaheurística en la resolución de la instancia Seattle

Por otro lado, la Figura 6.28 resume el estudio comparativo entre la hiperheurística paralela y siete equipos homogéneos construidos con la misma metaheurística cuando son aplicados a la resolución de la instancia Denver.

Como se puede observar, en este caso también se produce una mejora bastante significativa en cualquier periodo de tiempo. Además, en ambos casos se puede apreciar que las metaheurísticas que obtenían mejores resultados para ambas instancias también son los equipos homogéneos que consiguen planificaciones de frecuencia de mejor calidad (GA y VNS para el caso de Seattle, y SS y GA para el caso de Denver).

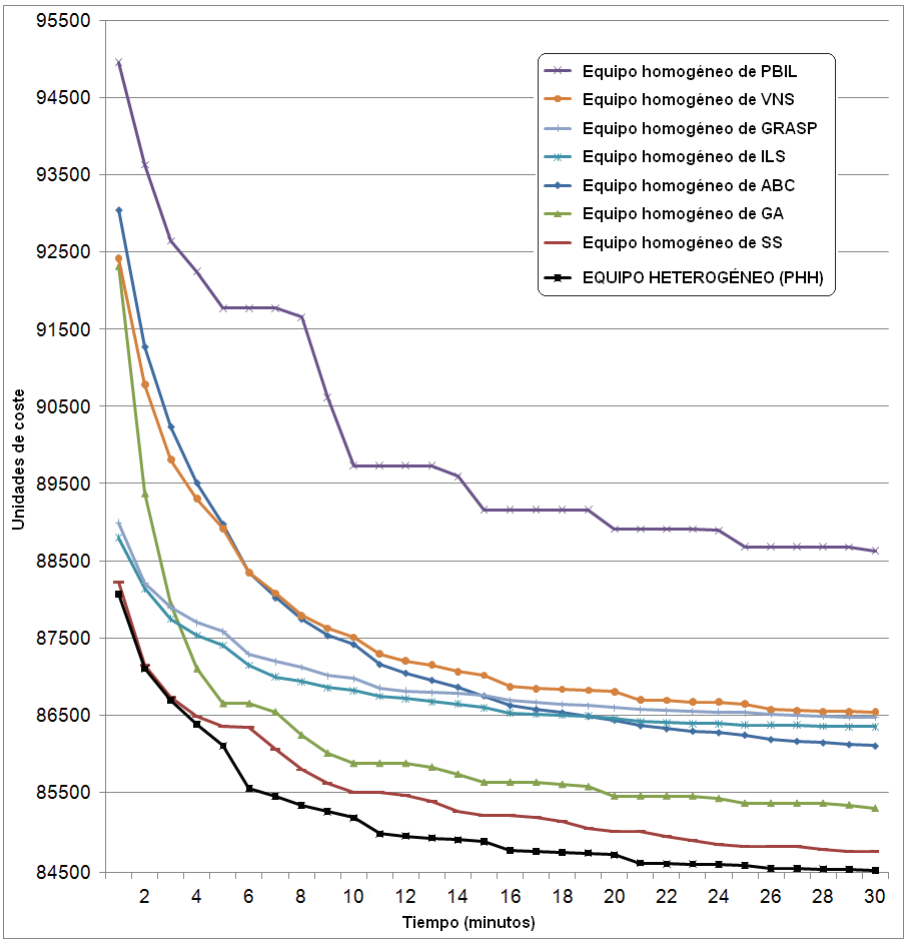

Figura 6.28. Comparativa de resultados entre la hiperheurística paralela (PHH, equipo heterogéneo) y siete equipos homogéneos formados por la misma metaheurística en la resolución de la instancia Denver

Por tanto, la hiperheurística paralela consigue unos resultados de muy alta calidad aprovechando las características de un conjunto heterogéneo de metaheurísticas que por separado obtienen resultados más pobres. La razón fundamental es que los algoritmos se complementan entre sí, por lo que en las situaciones en las que ciertas metaheurísticas se muestran más débiles, otras se muestran fuertes (y al contrario). Así, algunas metaheurísticas funcionan muy

bien para la instancia más grande (SS, GA y ABC, Figura 6.28), mientras que otras consiguen resultados mejores para la instancia más pequeña (PBIL, VNS, GA, Figura 6.27). Algunas son muy rápidas en su evolución (VNS, ILS y GRASP) mientras que otras consiguen resultados de muy alta calidad en ejecuciones más largas (GA y SS). Por tanto, se ha tratado de diseñar un equipo bien balanceado en el que la hiperheurística tuviera un amplio abanico de métodos entre los que elegir.

Pero además existen otros datos que justifican la elección de las metaheurísticas incluidas en el sistema paralelo, ya que por ejemplo PBIL es el algoritmo que peores resultados ofrece para la instancia Denver (Figura 6.26), y sin embargo el mismo algoritmo con la misma configuración ha sido muy relevante en la obtención de soluciones de alta calidad como parte de la PHH en el caso de la instancia Seattle (Figura 6.24.b). Es por esto que tras el análisis de los resultados se puede inferir que todas las metaheurísticas realizan alguna aportación importante en algún momento de la ejecución del sistema paralelo para alguna de las instancias abordadas. En resumen, se puede concluir que todas las metaheurísticas obtienen planificaciones de frecuencia de alta calidad para las dos instancias abordadas (Figuras 6.25 y 6.26), y que la hiper-heurística ha sido diseñada con un conjunto bien balanceado de metaheurísticas, ya que todas ellas contribuyen (Figura 6.24) en la obtención de soluciones de muy alta calidad del problema FAP.

Comparación con otros autores

En esta sección se comparan los resultados obtenidos en esta tesis con otros que han sido publicados en la bibliografía por otros autores. En algunos de los trabajos colaboró el autor de esta tesis, desarrollando alguno de los algoritmos incluidos en nuestra investigación. Los trabajos con los que se compara en este capítulo están sujetos a los mismos requerimientos y consideraciones que los presentados en esta tesis, por lo que sus resultados representan una fuente de comparación excepcional. La Tabla 6.9 presenta diferentes resultados obtenidos con diversas técnicas aplicadas a la resolución de la instancia Denver. Los

resultados finales conseguidos por la hiperheurística paralela (PHH) desarrollada en esta tesis se presentan en la primera fila de la tabla. Las siguientes filas contienen los datos arrojados por otros estudios relevantes (se incluye su referencia correspondiente).

Tabla 6.9. Comparación entre los resultados empíricos obtenidos por la PHH y los resultados obtenidos por otros estudios relevantes publicados en la bibliografía. Todos los trabajos resuelven la instancia Denver teniendo en cuenta las mismas consideraciones y restricciones explicadas en esta tesis. Cada aproximación incluye el mejor, la media y la desviación estándar de 30 ejecuciones independientes

	120 segundos		600 segundos		1800 segundos	
	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$	Mejor	$\bar{x}\pm\sigma$
PHH	**86640.3**	**87100.2±381.2**	**84246.6**	**85167.9±382.9**	**83815.8**	**84527.4±404.7**
ACO [119]	91140.0	93978.2±1165.9	89703.4	91726.4±1002.9	88345.9	90382.5±935.3
EA [119]	104247.7	108071.9±1723.4	100701.2	103535.9±1939.7	96490.5	99862.3±1553.1
GRASP [56]	88857.4	91225.7±1197.2	87368.4	89369.6±1185.1	86908.4	88850.6±1075.2
GridGRASP [56]	85313.0	87256.9±2309.2	85313.0	86772.1±1701.0	85259.4	85855.3±686.9
ACO [120]	90736.3	93439.5±1318.9	89946.3	92325.4±1092.8	89305.9	90649.9±727.5
ssGA [120]	87477.3	89540.4±991.1	86755.7	87850.8±573.6	85720.3	86908.9±379.8
SS [120]	91216.7	94199.6±1172.3	91069.8	93953.9±1178.6	91069.8	93820.4±1192.3
(1+2)EA [120]	87763.9	92294.0±1407.6	86064.8	89669.8±1164.8	85607.3	88574.3±1100.3
LSHR [120]	88543.0	92061.7±585.3	88031.0	89430.9±704.2	87743.0	88550.3±497.0
SS [127, 125]	86169.4	88692.7±1124.9	84570.6	86843.8±950.5	84234.5	85767.6±686.3
DE [127]	92145.8	95414.2±1080.4	89386.4	90587.2±682.3	87845.9	89116.8±563.8
Grid EA [141]	Resultados obtenidos en 1,5 horas: Mejor = 83991.3, $\bar{x}\pm\sigma$ = 84936.3±375.8					

Para cada estudio referenciado en la Tabla 6.9 se incluye el mejor y el resultado medio (junto con la desviación estándar) de 30 ejecuciones independientes en tres límites temporales distintos (120, 600 y 1800 segundos). Únicamente el trabajo que aparece en la última fila de la tabla (GridEA [141]) utiliza un límite temporal diferente al del resto de estudios, ya que el algoritmo desarrollado en ese estudio fue limitado por iteraciones en lugar de por un límite temporal. Sin embargo, el estudio se ha incluido en la tabla comparativa porque, hasta donde nosotros sabemos, representaba el mejor resultado publicado en la bibliografía en la resolución de la instancia Denver hasta que fue superado por los resultados obtenidos con la hiperheurística paralela explicada en esta tesis. Como se puede observar en la Tabla 6.9 incluso esos resultados (GridEA [141]), que fueron obtenidos después de 1,5 horas de ejecución, han sido superados por los resultados conseguidos por la PHH desarrollada en esta tesis, pero en un

tercio del tiempo invertido por el sistema grid (30 minutos), y utilizando menos recursos computacionales. El resto de estudios de la Tabla 6.9 utilizan los mismos límites temporales y condiciones experimentales que los utilizados en esta tesis.

Como se puede observar, el problema FAP ha sido abordado por diferentes técnicas metaheurísticas y aproximaciones paralelas en los últimos años (ver Capítulo 2: *Antecedentes y trabajo relacionado*). En esta tesis se han incluido algunas de estas técnicas metaheurísticas [56, 125] y se compara con los trabajos paralelos que utilizan las mismas instancias que las abordadas en nuestra investigación [56, 141]. La PHH explicada en esta tesis supera los resultados obtenidos por cualquier aproximación en cualquier límite temporal. Únicamente la aproximación basada en grid con la metaheurística GRASP (GridGRASP [56]), que también fue desarrollada como parte de nuestra investigación, consigue un mejor resultado para el caso *Mejor* en 120 segundos (ver Tabla 6.9), sin embargo el resultado medio de las 30 ejecuciones es peor que el conseguido por la PHH, y la desviación estándar es bastante alta, por lo que se puede considerar a ese resultado como un valor de pico del algoritmo.

6.6. Resumen

En este capítulo se muestran y discuten los resultados obtenidos con los experimentos más importantes que se han llevado a cabo con las técnicas abordadas en esta tesis. Todas las pruebas han sido realizadas utilizando los mismos recursos y sobre las mismas instancias del problema FAP.

En la primera parte del capítulo se han discutido los experimentos llevados a cabo con cada metaheurística, para después explicar las pruebas más importantes realizadas con la hiperheurística paralela diseñada a partir de éstas. Como conclusión, se puede destacar que las metaheurísticas que mejores resultados medios obtienen en la resolución del problema FAP son los algoritmos GA, SS, y VNS, si bien todas las técnicas ofrecen resultados de alta calidad, y realizan una aportación positiva al sistema paralelo diseñado con posterioridad.

De hecho, el coste que se desprende de las planificaciones de frecuencia generadas con todas las técnicas desarrolladas es bastante satisfactorio.

En la última parte del capítulo se ha realizado una comparativa con otros autores que también han trabajado con el mismo problema que se aborda en esta tesis. De esta comparativa se puede obtener una doble conclusión. Por un lado, tanto las metaheurísticas secuenciales desarrolladas como la aproximación paralela diseñada mejoran los resultados obtenidos en la mayoría de los trabajos publicados en la bibliografía, mientras que por otro lado se puede concluir que la hiperheurística paralela obtiene mejores resultados que cualquiera de los trabajos publicados de los cuales se tiene conocimiento.

Capítulo 7

Conclusiones

En esta tesis se han utilizado técnicas metaheurísticas y estrategias basadas en paralelismo para resolver un complejo problema de optimización dentro del ámbito de las telecomunicaciones. Se ha utilizado información procedente de la industria para el modelado del problema e instancias reales para realizar los experimentos. Con estos requisitos, la tarea de obtener soluciones óptimas para el problema supone una gran complejidad que conlleva enormes cantidades de cómputo. Las técnicas metaheurísticas en combinación con estrategias de paralelismo son los paradigmas que mejor se adaptan al problema abordado y los requisitos que deben ser satisfechos, ya que garantizan conseguir soluciones de muy alta calidad en tiempos razonables, lo cual es de gran importancia cuando se trabaja con problemas reales cuyos resultados deben ser aplicados a la industria.

El problema abordado en esta tesis es la planificación automática de frecuencias, que consiste en asignar el limitado espectro de frecuencias de que disponen las redes de comunicaciones al elevado conjunto de transmisores-receptores que se encargan de dar cobertura a toda la red, de manera que se establezca el servicio con la mayor calidad posible, es decir, reduciendo las interferencias que se producen en las comunicaciones a su mínima expresión. Para llevar a cabo esta tarea se ha utilizado un preciso modelo matemático que representa los requisitos del problema de manera fidedigna. A continuación se ha analizado, desarrollado y optimizado un conjunto diverso de metaheurísticas para dar solución eficiente y de calidad al problema anteriormente mencionando. Algunas de las técnicas utilizadas no habían sido nunca empleadas en la resolución de este problema, y otras, que sí lo habían sido, han sido mejoradas

de manera visible obteniéndose resultados de alta calidad para las instancias reales abordadas. Además, se han estudiado diversos modelos paralelos, algunos de los cuales han sido utilizados para agilizar el ajuste de los algoritmos desarrollados, mientras que otros han sido diseñados para mejorar los resultados de las aproximaciones secuenciales. En este sentido, se ha construido una hiper-heurística paralela que, haciendo uso de siete metaheurísticas, consigue aprovechar al máximo los recursos computacionales y algorítmicos de que hace uso, y obtiene planificaciones de frecuencia de muy alta calidad. Por tanto, los resultados expuestos en esta tesis revelan que los algoritmos y técnicas seleccionadas y desarrolladas para resolver el problema de la asignación de frecuencias son muy adecuados, ya que a parte de su teórica adecuación para resolver este tipo de problemas, con ellos se obtienen resultados de calidad que representan soluciones satisfactorias en entornos industriales.

En las siguientes secciones de este capítulo se exponen, en primer lugar, las principales aportaciones de esta tesis, a continuación el estado del trabajo actual y la investigación futura que se puede realizar como continuación de los resultados presentados en este documento, y finalmente, las conclusiones personales del autor de esta tesis.

7.1. Principales aportaciones

En esta tesis se ha realizado el estudio, modelado y ajuste de diversas técnicas metaheurísticas para conseguir resultados de alta calidad en la resolución del problema de la asignación automática de frecuencias en redes reales de comunicaciones. Los esquemas básicos de dichos algoritmos han sido modificados mediante la hibridación de dichas técnicas con otras heurísticas más sencillas para mejorar los resultados conseguidos por cada una de ellas en la resolución de instancias reales del problema FAP. Además, como ya se ha señalado anteriormente, algunas de las metaheurísticas con las que se ha trabajado en esta tesis no habían sido utilizadas antes en la resolución del problema abordado, y más aún, en un caso real del mismo, por lo que la

aplicación de esos algoritmos en la resolución de dicho problema es una contribución original de esta tesis doctoral.

Por otro lado, se ha diseñado e implementado una bastante novedosa estrategia paralela (hiperheurística paralela basada en metaheurísticas, PHH) que no había sido utilizada con anterioridad en la resolución del problema de la asignación automática de frecuencias en redes reales de telecomunicaciones. Hasta donde sabemos, la PHH desarrollada mejora los resultados (tanto en tiempo como en calidad) obtenidos hasta la fecha por cualquier otra aproximación diseñada para resolver el mismo problema que el que se aborda en esta tesis.

Asimismo, tanto el modelo matemático utilizado, como las instancias del problema con las que se han hecho todos los experimentos, están basados en datos reales provenientes de la industria de las telecomunicaciones (a través de la colaboración con otros investigadores dentro del proyecto OPLINK [14]), por lo que podemos decir que la investigación realizada y las conclusiones alcanzadas en esta tesis son aplicables a la industria del sector.

Finalmente, se ha procurado que los resultados y conclusiones que se han ido obteniendo durante el transcurso de esta investigación hayan ido siendo difundidos adecuadamente a la comunidad científica, por lo que la publicación de dichos avances en foros científicos de calidad puede considerarse una contribución importante aportada por esta tesis doctoral. A este respecto, el Apéndice II de esta tesis contiene la lista con las referencias de las publicaciones en las que se sustenta la investigación realizada.

7.2. Trabajo actual y futuro

En esta tesis se ha resuelto un problema complejo de optimización combinatoria utilizando técnicas metaheurísticas y estrategias basadas en paralelismo. El trabajo desarrollado en la misma ha servido para formar al doctorando tanto en el estudio y análisis de este tipo de problemas como en el diseño y desarrollo de técnicas apropiadas para su resolución. Se han estudiado y desarrollado diversas

metaheurísticas, tanto basadas en trayectoria como en población, y se han aplicado a un problema de optimización real dentro del campo de las telecomunicaciones. Asimismo, se han empleado diversas técnicas de paralelismo para agilizar los experimentos necesarios para ajustar las técnicas metaheurísticas anteriormente mencionadas y para mejorar los resultados obtenidos por éstas. Además, los resultados obtenidos se han difundido a través de diversos foros científicos de prestigio, lo cual proporciona validez al trabajo realizado. Sin embargo, el trabajo desarrollado en esta tesis es sólo el principio de la formación investigadora del autor de la misma. Hay numerosas líneas de trabajo futuro que surgen a partir de lo expuesto en este documento, algunas de las cuales supondrán los siguientes pasos que se realizarán y otras se dejarán como ideas que se desprenden del trabajo realizado y presentado en este documento. En el campo de la resolución del problema, por ejemplo, sería muy interesante realizar pruebas con más instancias del problema. Con respecto a las técnicas desarrolladas, aunque se ha hecho un estudio extenso de las mismas, éste podría ser ampliado con la utilización, por ejemplo, de metaheurísticas de reciente creación, como son los algoritmos Firefly [198], basado en el comportamiento de las luciérnagas, o GSA (*Gravitational Search Algorithm* [199]), basado en las leyes de la gravedad y de la interacción entre masas. En cuanto a las estrategias paralelas, se podría hacer un estudio mayor que el realizado en esta tesis, donde se aplicaran más técnicas de paralelismo, como por ejemplo un modelo de programación híbrida entre MPI [140] y openMP [181].

Otra línea de trabajo futuro que es de gran interés puede ser la aplicación de los algoritmos desarrollados a otros problemas de optimización, bien del mismo dominio o de otras disciplinas de la ingeniería. Esta será una de las líneas que se sigan en los próximos meses, ya que resulta de gran interés la aplicación y ampliación de las técnicas aprendidas en estos años a otros problemas complejos de optimización.

7.3. Conclusiones personales

El camino que ha supuesto la realización de esta tesis ha sido muy valioso en cuanto a experiencias y enriquecimiento personal. Este enriquecimiento va más allá de los estudios realizados y de las técnicas aprendidas. Durante los años de estudio de doctorado, el autor de esta tesis ha trabajado en diversas disciplinas, que van desde el desarrollo de simuladores con fines docentes o el estudio y desarrollo de sistemas de reconocimiento facial aplicando técnicas de tratamiento de imágenes, hasta el trabajo expuesto en esta tesis de resolución de un problema complejo de optimización dentro de ámbito de las telecomunicaciones haciendo uso de estrategias metaheurísticas y técnicas de paralelismo. Sin embargo, las disciplinas en las que se ha trabajado son sólo las ramas del árbol. El tronco lo conforman las vivencias y el desarrollo de la actividad científica, donde se ha aprendido, entre otras cosas, a buscar y analizar bibliografía específica, a estudiar y obtener información útil de artículos científicos y a verificar y cotejar datos procedentes de experimentos que pueden ser interesantes. También se ha tomado conciencia de que un trabajo científico, si bien puede ofrecer resultados de mucho interés, no puede ser considerado relevante hasta que los datos obtenidos no han sido cotejados y validados mediante una publicación científica de calidad que certifique su valía.

Por tanto, el enriquecimiento producido por la labor investigadora y todo lo aprendido durante estos años lleva mucho más allá del trabajo relatado en este documento. De esta forma, por ejemplo, la ardua tarea que supone la publicación de artículos, la asistencia a congresos, la participación en proyectos de investigación, el formar parte de un comité local de un congreso, o el proceso de revisión de artículos a otros compañeros, son tareas que han resultado tremendamente enriquecedoras personalmente, y como he indicado, van mucho más allá del desarrollo y análisis de técnicas metaheurísticas para resolver un problema de telecomunicaciones.

Apéndice I

Análisis estadístico de los resultados obtenidos en la tesis

En este apéndice se incluye un resumen del análisis estadístico realizado con los resultados obtenidos en esta tesis. En la primera parte del análisis se compararán los resultados obtenidos con las diferentes aproximaciones desarrolladas y estudiadas en esta tesis. El objetivo de dicho análisis es probar que las diferencias que presentan los resultados aportados por los distintos algoritmos son estadísticamente relevantes. Por otro lado, en la segunda parte se mostrará un resumen del análisis realizado de manera individual con los datos obtenidos en el ajuste de la configuración de las diferentes estrategias desarrolladas, de forma que se pruebe de manera formal que los experimentos que se han realizado son relevantes científicamente.

Desde el punto de vista estadístico, los resultados que se obtienen con las ejecuciones independientes que se realizan con cada estrategia pueden ser consideradas como muestras de una distribución de probabilidad. Todas las pruebas estadísticas han sido realizadas utilizando la herramienta *IBM SPSS Statistics 19* [200], siguiendo la explicación presentada en la sección 4.3 de esta tesis (*Evaluación estadística de las metaheurísticas*).

Análisis comparativo entre las distintas técnicas desarrolladas

En este apartado se investigan las diferencias entre los costes promedios asociados con las distintas aproximaciones desarrolladas para esta tesis. Se comparan, por tanto, los resultados de las 30 ejecuciones obtenidas por la hiperheurística paralela y por cada una de las metaheurísticas incluidas en nuestro estudio. El estudio se ha realizado utilizando de los datos obtenidos tras 30 minutos de ejecución independiente de cada uno de las aproximaciones. Los resultados medios pueden consultarse en la columna correspondiente a 1800 segundos de la Tabla 6.4 para la instancia Denver, así como de la misma columna de la Tabla 6.6 para la instancia Seattle. De igual forma, los mejores resultados de las 30 ejecuciones independientes realizadas pueden observarse en las mismas columnas de la Tabla 6.5 para Denver y 6.7 para Seattle. A continuación se muestra el estudio de los datos para la instancia Denver.

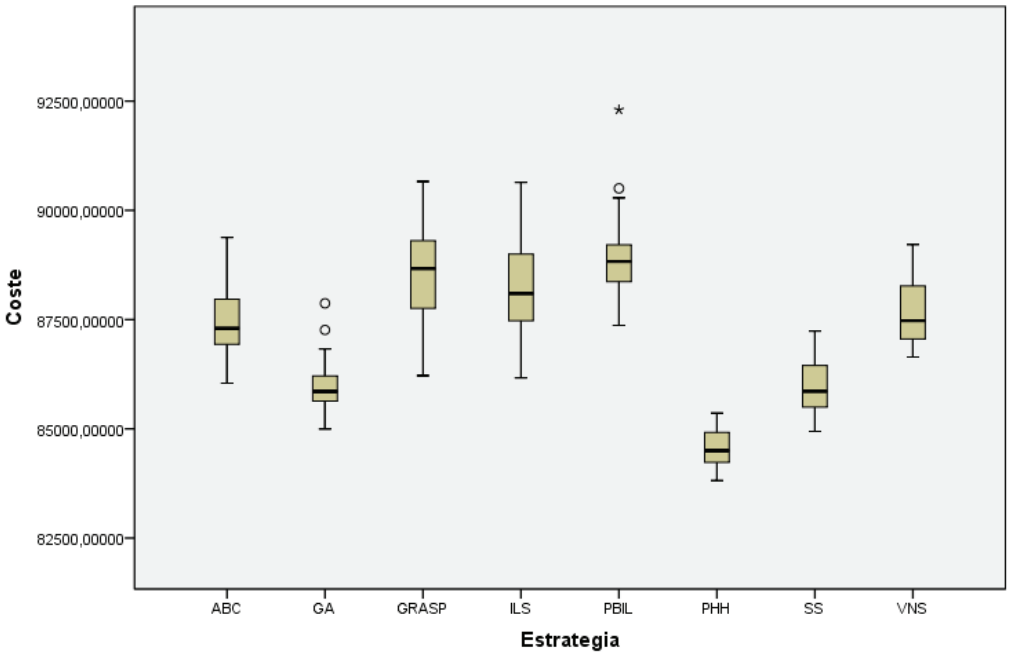

Figura AI.1. Diagrama de cajas para los resultados obtenidos para la instancia Denver tras 30 minutos con las distintas estrategias desarrolladas en la tesis

El primer resumen del análisis realizado consistió en un diagrama de cajas (Figura AI.1) con el que estudiar, para cada estrategia sometida a comparación, la distribución de los datos y la identificación de los casos atípicos. Como puede observarse, existen dos casos atípicos para PBIL y otras dos para el GA. Decidimos eliminar estos casos del estudio estadístico, considerándolos no representativos. Además, una observación muy importante es que la estrategia PHH ofrece los mejores resultados del diagrama: su coste mediana es el más bajo y es la más precisa (porque su dispersión es la más pequeña). En cualquier caso, una vez identificadas y eliminadas dichas muestras, estudiamos la dispersión del resto de datos, la cual se puede observar en la Figura AI.2, donde las estrategias aparecen ordenadas según su diseño (primero la estrategia paralela –PHH, luego las técnicas basadas en población –SS, GA, PBIL y ABC, y finalmente las basadas en trayectoria –ILS, VNS y GRASP).

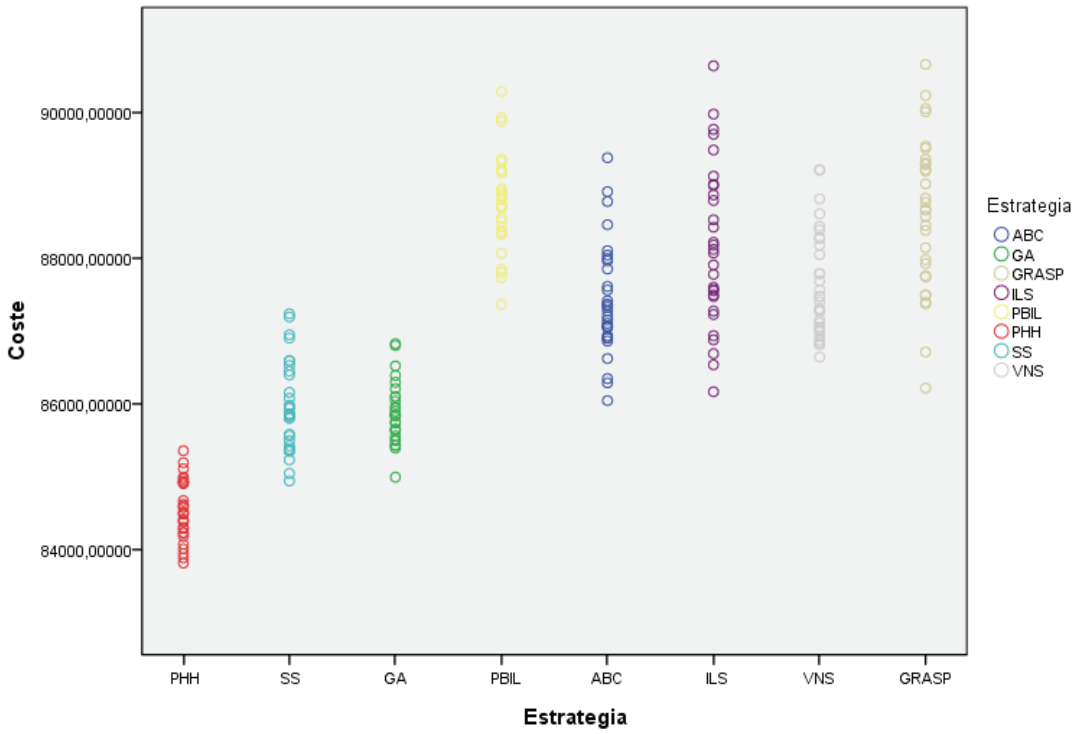

Figura AI.2. Diagrama de dispersión para los resultados obtenidos (sin los casos atípicos) en la resolución de la instancia Denver tras 30 minutos con las distintas estrategias desarrolladas

Como se puede observar, ninguna estrategia presenta una dispersión de muestras demasiado elevada, ya que los datos se desvían de la media un

máximo de 900 unidades de coste, lo cual no representa demasiada dispersión si se tiene en cuenta el valor de las medias del coste para cada estrategia. Una conclusión significativa es que el algoritmo PHH y el algoritmo GA son las estrategias que presentan menor dispersión. Por otro lado, el algoritmo GRASP y el ILS son los que generan unos datos con mayor dispersión, si bien ninguna aproximación presenta una dispersión muy elevada, por lo que se puede decir que los algoritmos proporcionan resultados bastante fiables de acuerdo con este parámetro.

El siguiente paso consiste en estudiar la normalidad de los residuos, mediante los test de Kolmogorov-Smirnov y Shapiro-Wilk (Tabla AI.1). La normalidad es uno de los requisitos básicos para poder aplicar el test de ANOVA. Como se puede observar, los resultados de las dos pruebas indican valores de significación superiores a 0,1 (columnas *Sig.* de la Tabla AI.1) para todas las estrategias desarrolladas, por lo que no se puede rechazar la hipótesis de normalidad en ningún caso. Luego a la vista de ambos test, es razonable suponer que los residuos asociados con las distintas estrategias desarrolladas forman muestras de distribuciones normales.

Tabla AI.1. Resultados de las pruebas de normalidad para la instancia Denver

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estrategia	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	ABC	,142	30	,123	,959	30	,285
	GA	,108	28	,200*	,966	28	,475
	GRASP	,081	30	,200*	,987	30	,969
	ILS	,096	30	,200*	,983	30	,889
	PBIL	,100	28	,200*	,981	28	,879
	PHH	,128	30	,200*	,974	30	,664
	SS	,132	30	,195	,960	30	,307
	VNS	,136	30	,165	,932	30	,054

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

El otro requisito que debe cumplirse para poder aplicarse el test de ANOVA es la homogeneidad de las varianzas. Esta característica se comprueba mediante el test de Levene (Tabla AI.2). Rechazamos la hipótesis de que las varianzas de los costes asociados con las distintas estrategias desarrolladas sean iguales, ya que el *p-value* (Sig.) del test de Levene es 0.000 (que es menor a 0.1). Esto

quiere decir que no es posible aplicar ANOVA, ya que no se cumple uno de los requisitos. En su lugar, se aplicará la prueba no paramétrica de Kruskal-Wallis (véase Figura 4.3), que investiga las diferencias entre las medianas, en lugar de las medias, como ANOVA. La Tabla AI.3 muestra el resultado de este test.

Tabla AI.2. Resultados del test de Levene para la instancia Denver

Prueba de homogeneidad de varianzas

Coste

Estadístico de Levene	gl1	gl2	Sig.
6,486	7	227	,000

Como se puede observar, tanto las medianas como las distribuciones que sigue el coste para las distintas estrategias desarrolladas son significativamente distintas (con un nivel de significación de 0.000), luego el análisis estadístico demuestra que la comparación entre los resultados aportados por las distintas estrategias en la resolución de la instancia Denver resulta relevante estadísticamente.

Tabla AI.3. Resultados del test de Kruskal-Wallis para los datos obtenidos en la resolución de la instancia Denver

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Estrategia.	Prueba de medianas de muestras independientes	,000	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Estrategia.	Prueba Kruskal-Wallis de muestras independientes	,000	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

A continuación se realiza el mismo estudio para los datos obtenidos en la resolución de la instancia Seattle. La Figura AI.3 muestra el diagrama de cajas con la distribución de los datos conseguidos por cada estrategia desarrollada. Al igual que con la instancia Denver, el primer paso consistió en identificar los casos atípicos de las distintas estrategias y eliminarlos. En este caso, se hallaron dos casos atípicos para el algoritmo VNS, con lo que el número de casos que se

han considerado para realizar el análisis de este algoritmo han sido finalmente 28. El algoritmo ILS presenta también dos casos atípicos que fueron detectadas y eliminadas, al igual que el GRASP. Finalmente, se detectó y eliminó un caso atípico para el algoritmo ABC. La dispersión de los datos resultantes, tras este estudio preliminar, puede observarse en la Figura AI.4. En este caso, también la PHH, junto con los algoritmos SS y GA son los que presentan menor dispersión.

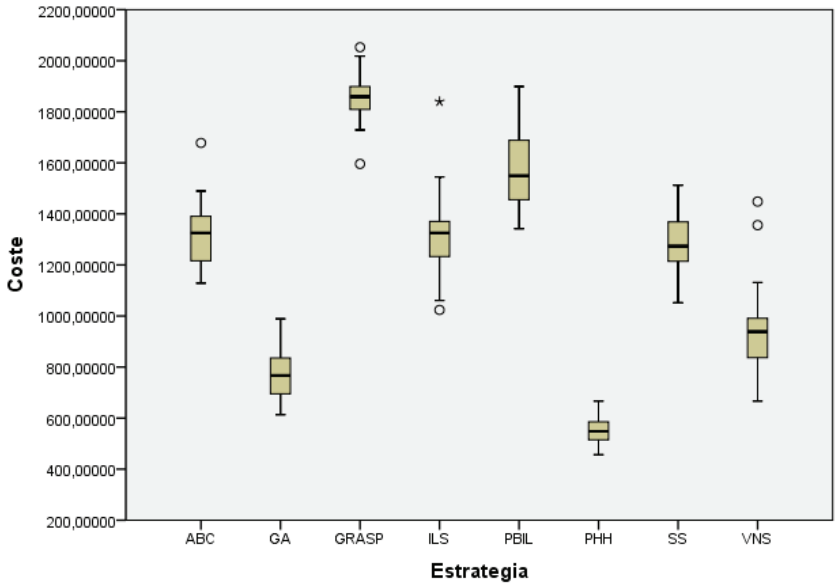

Figura AI.3. Diagrama de cajas para los resultados obtenidos para la instancia Seattle tras 30 minutos con las distintas estrategias desarrolladas en la tesis

Por tanto, la Figura AI.4 muestra un diagrama de dispersión de los resultados obtenidos por cada estrategia una vez eliminadas las muestras atípicas. Como se puede observar, en este caso la dispersión en las muestras de las 30 ejecuciones tampoco es demasiado grande (menor a 140 unidades de coste en el peor de los casos), siendo otra vez el algoritmo PHH el que presenta menor dispersión, luego datos más fiables atendiendo a esta observación.

Por otro lado, la Tabla AI.4 muestra el resultado del estudio de la normalidad de los residuos (pruebas de Kolmogorov-Smirnov y Shapiro-Wilk). Como se puede observar en dicha tabla, los residuos de todas las estrategias desarrolladas superan el test de normalidad de Shapiro-Wilk (columna *Sig*. de dicho test). En cuanto al de Kolmogorov-Smimov, únicamente el algoritmo ABC presenta un valor de significación inferior a 0,1 (*Sig*. = 0,066), pero este valor es superior a 0,05, lo cual indica cierto nivel de incertidumbre a la hora de poder

rechazar la hipótesis nula de la prueba (que es la distribución normal de los residuos), por lo que este hecho sumado al resultado del test de Shapiro-Wilk hace que podamos interpretar que los residuos son normales.

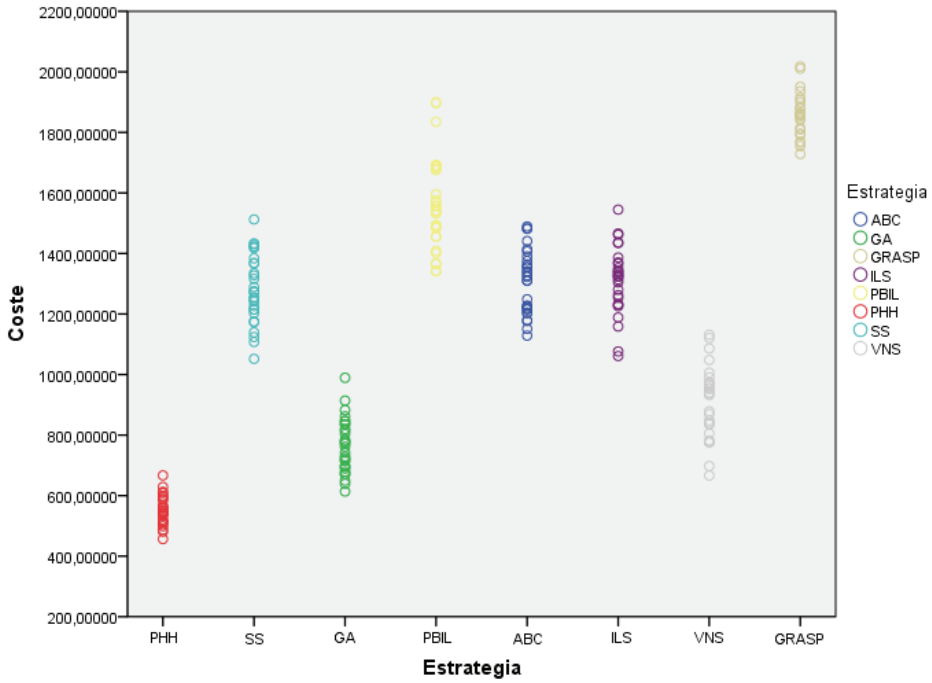

Figura AI.4. Diagrama de dispersión para los resultados obtenidos (sin los casos atípicos) en la resolución de la instancia Seattle tras 30 minutos con las distintas estrategias desarrolladas

Tabla AI.4. Resultados de las pruebas de normalidad para la instancia Seattle

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estrategia	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	ABC	,157	29	,066	,952	29	,211
	GA	,090	30	,200*	,977	30	,743
	GRASP	,121	28	,200*	,962	28	,384
	ILS	,116	28	,200*	,972	28	,630
	PBIL	,113	30	,200*	,944	30	,119
	PHH	,099	30	,200*	,986	29	,962
	SS	,094	30	,200*	,982	30	,886
	VNS	,121	28	,200*	,974	28	,694

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Por otro lado, la Tabla AI.5 muestra el resultado del test de Levene que comprueba la homogeneidad de las varianzas.

Tabla AI.5. Resultados del test de Levene para la instancia Seattle

Prueba de homogeneidad de varianzas

Coste

Estadístico de Levene	gl1	gl2	Sig.
5,504	7	224	,000

Como se puede observar, en el caso de los resultados producidos por la instancia Seattle, también rechazamos la hipótesis de que las varianzas de los costes asociados con las distintas estrategias desarrolladas sean iguales, ya que el *p-value* (Sig.) del test de Levene es 0,000. Esto quiere decir que para estos datos tampoco es posible realizar el estudio de las medias a través de ANOVA. En su lugar, se aplica la prueba no paramétrica de Kruskal-Wallis (véase Figura 4.3), que investiga las diferencias entre las medianas en lugar de las medias. La Tabla AI.6 muestra el resultado de este test.

Tabla AI.6. Resultados del test de Kruskal-Wallis para los datos obtenidos en la resolución de la instancia Seattle

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Estrategia.	Prueba de medianas de muestras independientes	,000	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Estrategia.	Prueba Kruskal-Wallis de muestras independientes	,000	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

Como se puede observar, en este caso las medianas y las distribuciones del coste para las distintas estrategias desarrolladas son también significativamente distintas (con un nivel de significación de 0,000), luego el análisis estadístico demuestra que la comparación entre los resultados aportados por las distintas estrategias en la resolución de la instancia Seattle también resulta relevante estadísticamente.

Análisis estadístico de cada estrategia desarrollada

En esta sección se incluye un resumen del análisis estadístico llevado a cabo con los datos obtenidos durante el ajuste de cada uno de los algoritmos que ha sido desarrollado en esta tesis. En la sección anterior se comprobó que la comparación entre las estrategias desarrolladas es relevante, ya que tanto las medianas como las distribuciones que siguen los costes obtenidos por cada algoritmo son estadísticamente significativas. La rodaja de tiempo que se ha elegido para realizar el análisis es 30 minutos, ya que este límite temporal ha sido el elegido tanto para realizar las comparaciones entre las distintas aproximaciones desarrolladas en esta investigación, como el utilizado en otros estudios relevantes publicados. Otras rodajas de tiempo inferiores, como 10 o 2 minutos presentan mayores diferencias entre las medias de los datos, y por tanto mayor diferenciación y relevancia estadística, pero los resultados obtenidos tras 30 minutos de ejecución son de mayor calidad, por lo que su validez y relevancia estadística representa mayor valor científico. El número de experimentos realizados con cada algoritmo, sobre todo con aquellos que se desarrollaron en las primeras etapas de nuestra investigación, ha sido muy elevado, por lo que en esta sección se incluye únicamente el análisis de los experimentos más relevantes realizados.

Análisis de los resultados obtenidos con la hiperheurística paralela

Los dos parámetros de configuración fundamentales de la hiperheurística paralela son el número de sincronizaciones que se realizan entre el proceso maestro y los esclavos, y el número de cores que se utilizan para la ejecución de las distintas metaheurísticas. En esta sección se incluye el análisis estadístico del ajuste de ambos parámetros para la instancia Denver y para la Seattle. En primer lugar se realizaron los experimentos para determinar el número de sincronizaciones óptimo para el sistema. El diagrama de cajas con el que se

analizó la dispersión de los datos respecto a la mediana para detectar y eliminar muestras atípicas del estudio puede observarse en la Figura AI.5. Como se puede observar únicamente se eliminó un dato correspondiente a la sincronización en todos los minutos de la ejecución del algoritmo.

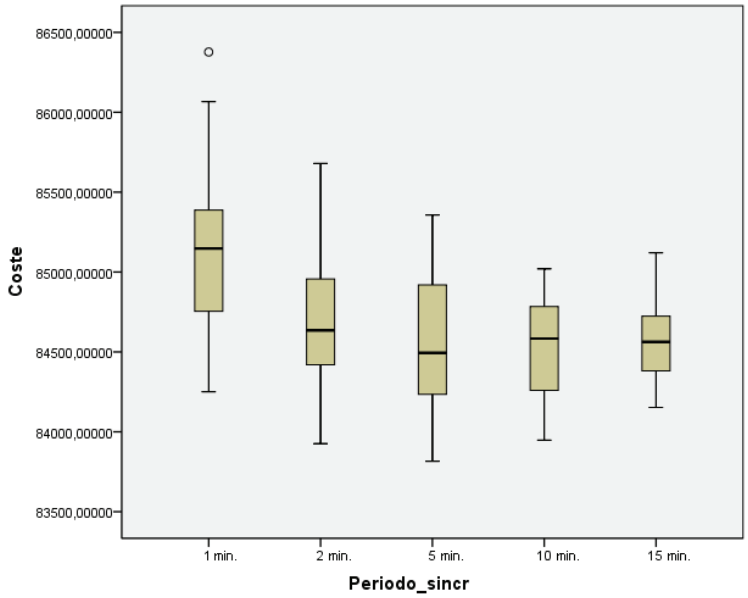

Figura AI.5. Diagrama de cajas para los resultados obtenidos tras 30 minutos con diferentes periodos de sincronización de la PHH para la instancia Denver

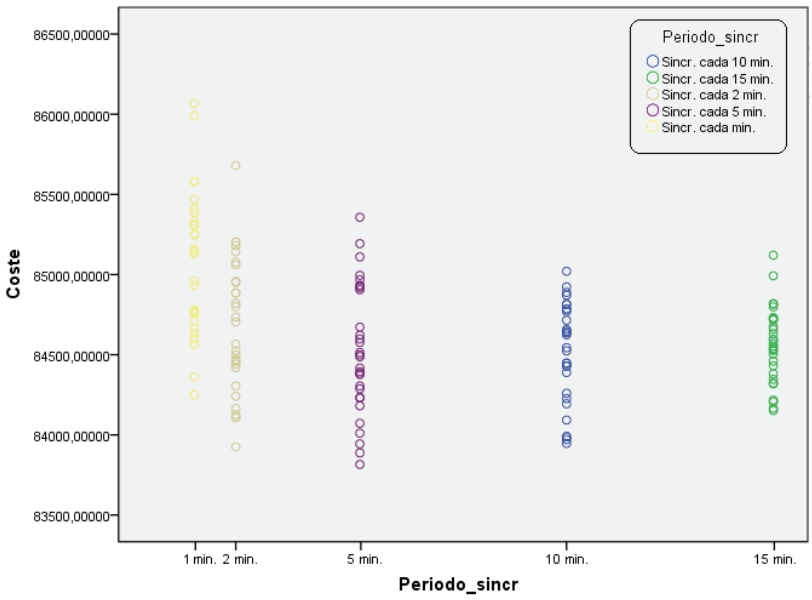

Figura AI.6. Diagrama de dispersión para los resultados obtenidos por la PHH en la resolución de la instancia Denver

Tras la eliminación de la muestra atípica (Figura AI.5), se realizó un estudio de la dispersión de las muestras para cada experimento mediante el correspondiente diagrama de dispersión (Figura AI.6). Como se puede observar, los datos presentan menor dispersión cuanto mayor es el periodo de sincronización, lo cual es normal ya que cuantas menos sincronizaciones estén configuradas entre el proceso maestro y los esclavos del sistema, menos aportaciones fuera de la evolución global se contemplarán. En cualquier caso, la dispersión general de los datos no es muy elevada en ningún experimento, por lo que los resultados pueden ser considerados fiables atendiendo a este parámetro. Las Tablas AI.7 y AI.8 muestran los resultados de las pruebas de normalidad de los residuos (Kolmogorov-Smirnov y Shapiro-Wilk) y de homogeneidad de las varianzas (Levene), respectivamente, que como se ha explicado anteriormente, son dos condiciones necesarias para poder realizar el test de ANOVA (Figura 4.3).

Tabla AI.7. Resultados de las pruebas de normalidad para la instancia Denver

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Periodo_sincr	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo	1	,137	29	,191	,971	29	,594
	2	,103	30	,200*	,972	30	,595
	5	,125	30	,200*	,974	30	,647
	10	,135	30	,173	,940	30	,092
	15	,076	30	,200*	,977	30	,754

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Tabla AI.8. Resultados del test de Levene para la instancia Denver

Prueba de homogeneidad de varianzas

Coste

Estadístico de Levene	gl1	gl2	Sig.
4,233	4	143	,003

Como se puede observar, el test de Kolmogorov-Smirnov (K-S) indica que para todos los experimentos, los residuos siguen una distribución normal (al ser su nivel de significación superior a 0.1 –columna Sig.). En cuanto al test de Levene, da un resultado negativo, ya que su nivel de significación es inferior a 0,1, por tanto, no se puede aplicar el test de ANOVA.

Por tanto, se aplicará la prueba no paramétrica de Kruskal-Wallis (K-W, Figura 4.3), que hace un estudio en base a las medianas, en lugar de en base a la media, como ANOVA. La Tabla AI.9 muestra el resultado de este test. Como se puede observar, tanto las medianas como las distribuciones que sigue el coste para los distintos experimentos realizados con las sincronizaciones de la PHH son significativamente distintas (con un nivel de significación de 0.05), luego el análisis estadístico demuestra que los resultados obtenidos en los experimentos realizados con la instancia Denver resulta relevante estadísticamente.

Tabla AI.9. Resultados del test de Kruskal-Wallis para los datos obtenidos en la resolución de la instancia Denver

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Periodo_sincr.	Prueba de medianas de muestras independientes	,001	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Periodo_sincr.	Prueba Kruskal-Wallis de muestras independientes	,000	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

A continuación se realiza el mismo estudio para los datos obtenidos en la resolución de la instancia Seattle. Se comienza mostrando el diagrama de cajas de los datos generados respecto a la mediana de los mismos (Figura AI.7). Como se puede observar, únicamente se detecta una muestra resulta atípica para el experimento de sincronizar cada 10 minutos. Tras su eliminación, el diagrama de dispersión de los distintos experimentos con la instancia Seattle es el mostrado en la Figura AI.8. A la vista de la dispersión de las muestras, se puede afirmar que en el caso de la instancia Seattle, los datos también presentan menor dispersión cuanto mayor es el periodo de sincronización, lo cual ya se comentó que se consideraba un comportamiento normal, ya que cuantas menos sincronizaciones estén configuradas, menos aportaciones atípicas habrá en la evolución global del sistema. De cualquier modo, en ningún caso la dispersión general de los datos es muy elevada, por lo que los resultados pueden ser considerados fiables atendiendo a este parámetro (Figuras AI.7 y AI.8).

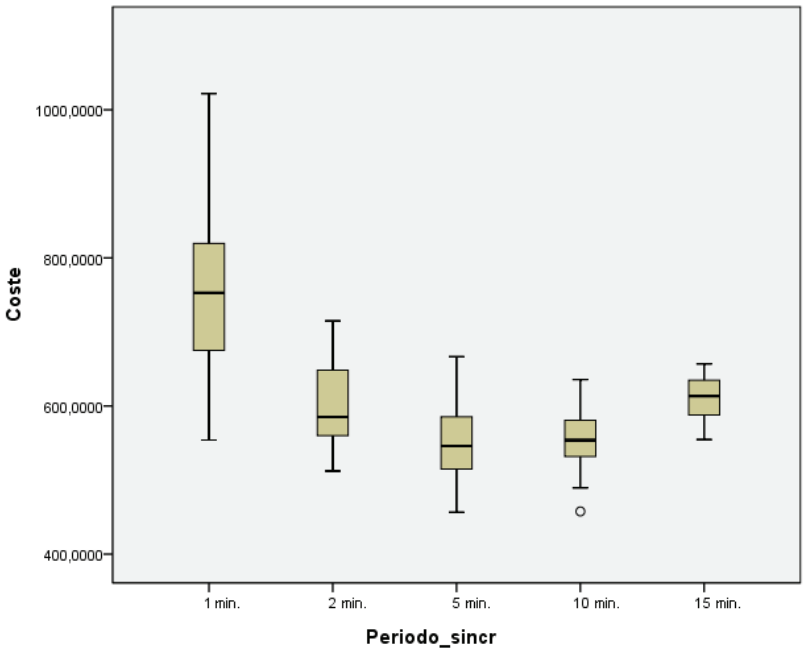

Figura AI.7. Diagrama de cajas para los resultados obtenidos tras 30 minutos con diferentes periodos de sincronización de la PHH para la instancia Seattle

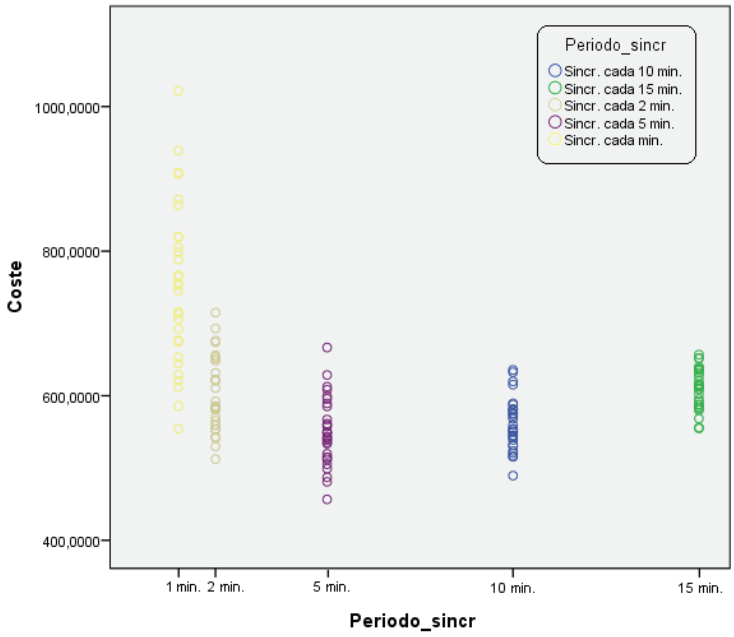

Figura AI.8. Diagrama de dispersión para los resultados obtenidos por la PHH en la resolución de la instancia Seattle

Las Tablas AI.10 y AI.11 muestran los resultados de las pruebas de normalidad de los residuos (Kolmogorov-Smirnov y Shapiro-Wilk) y de homogeneidad de las varianzas (Levene), respectivamente, que como se ha explicado anteriormente, son dos condiciones necesarias para poder realizar el test de ANOVA (Figura 4.3).

Tabla AI.10. Resultados de las pruebas de normalidad para la instancia Seattle

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Periodo_sincr	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	1	,073	30	,200*	,982	30	,893
	2	,156	30	,062	,965	30	,422
	5	,104	30	,200*	,984	30	,928
	10	,097	29	,200*	,972	29	,624
	15	,105	30	,200*	,968	30	,476

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Como se puede observar, el test de Shapiro-Wilk indica que para todos los experimentos, los residuos siguen una distribución normal (al ser su nivel de significación superior a 0.1 –columna Sig.). En cuanto al test de Levene, da un resultado negativo, ya que su nivel de significación es inferior a 0,1, por tanto, no se puede aplicar el test de ANOVA.

Tabla AI.11. Resultados del test de Levene para la instancia Seattle

Prueba de homogeneidad de varianzas

Coste

Estadístico de Levene	gl1	gl2	Sig.
5,504	7	224	,000

A la vista de los resultados de las pruebas, en este caso también se aplicará la prueba no paramétrica de Kruskal-Wallis (Figura 4.3). La Tabla AI.12 muestra el resultado de este test. Como se puede observar, tanto las medianas como las distribuciones que sigue el coste para los distintos experimentos realizados con las sincronizaciones de la PHH son significativamente distintas (con un nivel de significación de 0.000), luego el análisis estadístico demuestra que los resultados obtenidos en los experimentos realizados con la instancia Seattle también resultan relevantes estadísticamente.

Tabla AI.12. Resultados del test de Kruskal-Wallis para los datos obtenidos en la resolución de la instancia Seattle

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Periodo_sincr.	Prueba de medianas de muestras independientes	,000	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Periodo_sincr.	Prueba Kruskal-Wallis de muestras independientes	,000	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

Análisis de los resultados obtenidos con las metaheurísticas

En esta sección se incluyen las conclusiones del análisis estadístico llevado a cabo con cada una de las metaheurísticas desarrolladas en esta tesis. No se ha incluido un análisis tan exhaustivo como el expuesto en los apartados anteriores, ya que el ajuste de estos algoritmos requirió de un número muy elevado de experimentos en muchos casos. Por esta razón se incluyen las conclusiones de las pruebas más importantes, las cuales sirven para probar estadísticamente la validez de los resultados obtenidos.

Análisis de los resultados del algoritmo ILS

El diagrama de dispersión para los resultados obtenidos tras 30 minutos en los diferentes experimentos que se hicieron para ajustar el parámetro probabilidad de mutación puede observarse en la Figura AI.9. Como se puede observar, los datos no presentan una dispersión muy elevada, por lo que puede considerarse que son fiables atendiendo a este parámetro.

El siguiente paso consiste en estudiar la normalidad de los residuos, mediante los test de Kolmogorov-Smirnov y Shapiro-Wilk (Tabla AI.13). La

normalidad es uno de los requisitos básicos para poder aplicar el test de ANOVA. Como se puede observar, no todos los experimentos presentan una distribución residual normal. En concreto, los datos para las probabilidades de mutación igual a 0,4 y a 0,8, presentan una distribución que no es normal, ya que el nivel de significación de las muestras es menor a 0,1 para los dos test aplicados (véanse columnas *Sig.* de la Tabla AI.13). Por esta razón, no es posible aplicar el test de ANOVA con los datos correspondientes a los experimentos realizados con el algoritmo ILS. En su lugar, se utilizará la prueba no paramétrica de Kruskal-Wallis, cuyo resultado puede apreciarse en la Tabla AI.14.

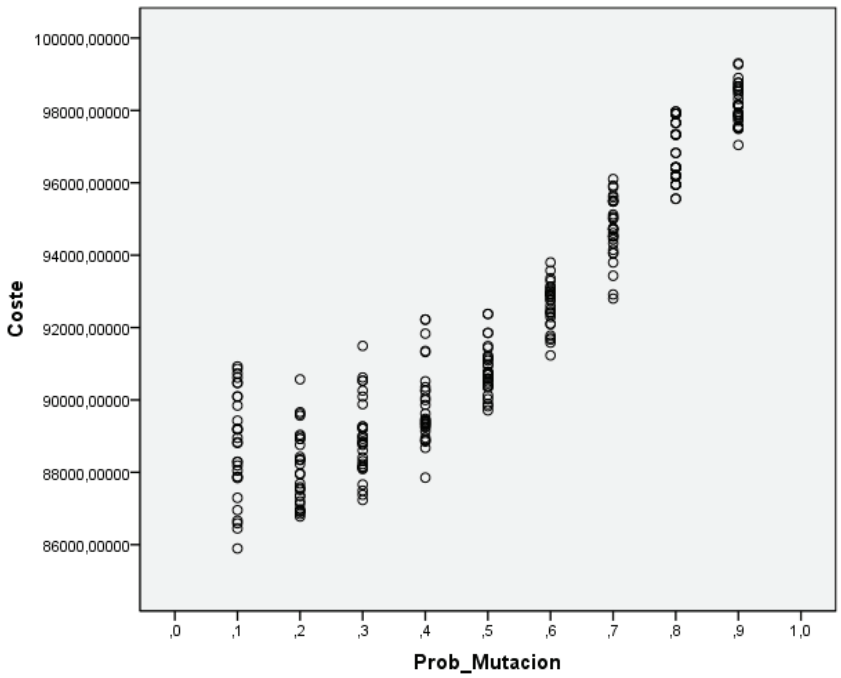

Figura AI.9. Dispersión de los datos obtenidos en los experimentos para ajustar la probabilidad de mutación en el algoritmo ILS

Tabla AI.13. Pruebas de normalidad para los experimentos realizados con el objetivo de ajustar la probabilidad de mutación en el algoritmo ILS

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Prob_Mutacion	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	,1	,096	30	,200*	,957	30	,275
	,2	,121	30	,200*	,937	30	,075
	,3	,146	30	,103	,960	30	,308
	,4	,193	30	,006	,904	30	,011
	,5	,135	30	,171	,955	30	,227
	,6	,147	30	,095	,963	30	,377
	,7	,110	30	,200*	,959	30	,301
	,8	,212	30	,001	,904	30	,011
	,9	,105	30	,200*	,975	30	,672

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Tabla AI.14. Resultado del test de K-W para los experimentos realizado con la probabilidad de mutación del algoritmo ILS en la resolución de la instancia Denver

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Prob_Mutacion.	Prueba de medianas de muestras independientes	,000	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Prob_Mutacion.	Prueba Kruskal-Wallis de muestras independientes	,000	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

Como se puede observar, tanto las medianas como las distribuciones que sigue el coste para los distintos experimentos realizados son significativamente distintas (con un nivel de significación de 0.000, como se aprecia en la Tabla AI.14), por lo que los experimentos llevados a cabo para ajustar la probabilidad de mutación del algoritmo ILS pueden considerarse estadísticamente relevantes.

Análisis de los resultados del algoritmo VNS

Los experimentos principales que se realizaron con el algoritmo VNS fueron los relativos para ajustar los parámetros k_{max} y α. Las dispersiones de las muestras

para los resultados obtenidos de los experimentos que se realizaron para ajustar ambos parámetros no fueron demasiado altas. Por otro lado, en este caso sí que fue posible utilizar el test de ANOVA, ya que los datos cumplen tanto la normalidad de los residuos como la homocedasticidad de las varianzas. En las Tablas AI.15 y AI.16 se muestra el resultado de las pruebas de normalidad de los residuos para ambos parámetros, mientras que en la Tabla AI.17 se puede ver el resultado de las pruebas de homogeneidad de las varianzas.

Tabla AI.15. Pruebas de normalidad para los experimentos realizados con el objetivo de ajustar el parámetro k_{max} en el algoritmo VNS

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	kmax	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	1	,096	30	,200*	,921	30	,028
	2	,171	30	,025	,943	30	,113
	3	,086	30	,200*	,978	30	,782
	4	,103	30	,200*	,969	30	,513
	5	,135	30	,174	,963	30	,360
	6	,101	30	,200*	,963	30	,360
	7	,137	30	,160	,923	30	,032
	8	,095	30	,200*	,965	30	,415
	9	,082	30	,200*	,971	30	,578

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Tabla AI.16. Pruebas de normalidad para los experimentos realizados con el objetivo de ajustar el parámetro α en el algoritmo VNS

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	alfa	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	,0	,115	30	,200*	,955	30	,236
	,2	,103	30	,200*	,964	30	,399
	,4	,119	30	,200*	,955	30	,244
	,6	,091	30	,200*	,979	30	,770
	,8	,118	30	,200*	,922	30	,069
	1,0	,114	30	,200*	,936	30	,072

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

A la vista de ambos test, es razonable suponer que los residuos asociados con los distintos experimentos de ajuste llevados a cabo forman muestras de distribuciones normales, ya que la significación de todas las pruebas (columnas

Sig. en las Tablas AI.15 y AI.16) resultó ser mayor a 0,1 para, al menos, uno de los dos test realizados.

Tabla AI.17. Pruebas de homogeneidad de las varianzas para los experimentos realizados en el ajuste del parámetro α (alfa) y k_{max} (k_max)

Prueba de homogeneidad de varianzas para el parámetro alfa

Coste

Estadístico de Levene	gl1	gl2	Sig.
,904	5	174	,480

Prueba de homogeneidad de varianzas para el parámetro k_max

Coste

Estadístico de Levene	gl1	gl2	Sig.
1,392	8	261	,200

En cuanto a la homogeneidad de las varianzas, como se puede observar, el nivel de significación (columnas Sig. de la Tabla AI.17) es de 0,480 en el caso de los datos provenientes de los experimentos con el parámetro α y de 0,200 en el caso de los datos provenientes del ajuste del parámetro k_{max}, por lo que no se puede rechazar la hipótesis de que las varianzas de los costes asociados en ambos conjuntos de experimentos sean iguales.

Por tanto, al cumplirse los requisitos para que pueda aplicarse el test de ANOVA de una vía, se realizó dicho test, obteniéndose los resultados que se muestran en la Tabla AI.18 para el caso del parámetro k_{max} y Tabla AI.19 para el caso del parámetro α.

Tabla AI.18. Resultado del test de ANOVA para el estudio de las medias del parámetro k_{max}

Coste

ANOVA

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Inter-grupos	74956073,75	8	9369509,219	4,936	,000
Intra-grupos	4,954E8	261	1898062,209		
Total	5,704E8	269			

Tabla AI.19. Resultado del test de ANOVA para el estudio de las medias del parámetro α

Coste

ANOVA

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Inter-grupos	1,251E8	5	25027784,24	23,423	,000
Intra-grupos	1,859E8	174	1068528,690		
Total	3,111E8	179			

Como se puede observar en ambos casos (Tablas AI.18 y AI.19), el valor de la significación (columnas Sig.) para la comparación de las medias de los distintos experimentos realizados es 0,000, por lo que no rechazamos la hipótesis nula de que las medias sean iguales. Esto quiere decir que las medias de los resultados que se obtienen con los experimentos de ajuste del algoritmo VNS son significativamente distintas.

Análisis de los resultados del algoritmo GRASP

El algoritmo GRASP generó una gran cantidad de resultados, ya que se hicieron pruebas utilizando una infraestructura grid con las 4 variantes del algoritmo (cardinalidad, valor, RG y PG), con el valor del *k* y α apropiado en cada caso, y tres tipos de sesgo distinto (aleatorio, lineal y exponencial) en aquellas versiones donde es posible su aplicación. Al utilizar computación grid, se realizaron todas las pruebas posibles, resultando una gran cantidad de experimentos. En esta sección, se incluye a modo de resumen el análisis estadístico de las mejores configuraciones obtenidas para cada variante, para demostrar que las diferencias presentes entre cada una de ellas son relevantes estadísticamente.

En primer lugar, igual que en los casos anteriores, se analizó la dispersión de los datos obtenidos con cada versión desarrollada del algoritmo. Al igual que en el resto de algoritmos estudiados, la dispersión de los datos generados con los experimentos que se realizaron con cada una de las versiones implementadas del GRASP, no resulta ser muy elevada en ningún caso, por lo que se puede considerar a las muestras obtenidas en dichos experimentos estadísticamente representativas. El siguiente paso del análisis consistió en

evaluar la normalidad de los residuos de las muestras mediante los test Kolmogorov-Smirnov y Shapiro-Wilk. Tras un análisis exhaustivo de los datos generados, se puede concluir que no se cumple la normalidad de los residuos de manera general (Tabla AI.20), por lo que no es posible realizar una comprobación de la relevancia de los resultados obtenidos en base al estudio de la media mediante el test de ANOVA. En su lugar, será necesario realizar una prueba no paramétrica para muestras independientes, como es el test de Kruskal-Wallis, que comprueba la relevancia estadística de los datos con relación a la mediana.

Tabla AI.20. Resultados de las pruebas de normalidad residual para distintas versiones del algoritmo GRASP

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Cod_Exp	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Residuo para Coste	1	,060	30	,200*	,986	30	,954
	2	,212	30	,002	,898	30	,010
	3	,133	30	,182	,979	30	,790
	4	,130	30	,200*	,963	30	,377

a. Corrección de la significación de Lilliefors
*. Este es un límite inferior de la significación verdadera.

Como se puede observar en la Tabla AI.20, no se puede garantizar la normalidad de los residuos para algunos datos. En concreto, el experimento de la segunda fila, que representa un ejemplo con la versión 2, presenta un *p-value* inferior a 0,1 para las dos pruebas de normalidad utilizadas. Este hecho se repite en más experimentos de todas las versiones del GRASP estudiadas, por lo que no es posible utilizar el test de ANOVA al no cumplirse uno de los requisitos de la prueba.

Por tanto, se utilizó la prueba de Kruskal-Wallis para demostrar la relevancia estadística de los datos generados con los distintos experimentos a los que se sometió el algoritmo GRASP. Este test sí resultó ser positivo para todos los casos comprobados, dando un resultado como el que se muestra en la Tabla AI.21.

Tabla AI.21. Resultado del test de Kruskal-Wallis para cuatro experimentos realizados con cuatro versiones diferentes del algoritmo GRASP

Resumen de prueba de hipótesis

	Hipótesis nula	Test	Sig.	Decisión
1	Las medianas de Coste son las mismas entre las categorías de Cod_Exp.	Prueba de medianas de muestras independientes	,004	Rechazar la hipótesis nula.
2	La distribución de Coste es la misma entre las categorías de Cod_Exp.	Prueba Kruskal-Wallis de muestras independientes	,018	Rechazar la hipótesis nula.

Se muestran las significancias asintóticas. El nivel de significancia es .05.

Por tanto, se puede concluir que tanto las medianas como la distribución del coste de los resultados obtenidos en los distintos experimentos del algoritmo GRASP que fueron comprobados, pueden considerarse diferentes, ya que en todos los casos se rechaza la hipótesis nula del test (columna Sig. < 0,1), por lo que los experimentos llevados a cabo con dicho algoritmo pueden considerarse estadísticamente relevantes en base a las pruebas realizadas.

Análisis de los resultados del algoritmo PBIL

El algoritmo PBIL, como el resto de las metaheurísticas basadas en población, incluye numerosos parámetros que deben ajustarse al problema al que el algoritmo está siendo aplicado para conseguir que éste genere resultados óptimos. El número de experimentos para llevar a cabo este ajuste es muy elevado, por lo que en este apartado se incluye únicamente un resumen del análisis estadístico al que fueron sometidos los datos provenientes de los experimentos de ajuste con el algoritmo.

Como en los casos anteriores, lo primero que se realizó tras obtener el resultado de cada experimento fue un estudio de la dispersión de las muestras obtenidas. Al igual que para los demás algoritmos, las muestras presentan una baja dispersión, por lo que los datos pueden considerarse fiables atendiendo a este parámetro.

A continuación se realizó un análisis estadístico que validara la relevancia estadística de los diferentes experimentos a través de diferentes test. El primer intento de validación de los datos se realiza siempre con una prueba estándar de gran potencia, el test de ANOVA, que certifica que los datos sometidos a estudio presentan medias significativamente distintas. Sin embargo, para poder aplicar este test se han de cumplir algunas condiciones, como se ha mencionado anteriormente. Caso de no poderse aplicar ANOVA, los datos serán analizados utilizando la prueba no paramétrica de Kruskal-Wallis. En los siguientes párrafos se resumen los resultados y conclusiones obtenidas tras aplicar los citados test a los datos provenientes del ajuste paramétrico del algoritmo PBIL para la rodaja de tiempo de 30 minutos, que es la que se ha seleccionado como objeto de estudio para el estudio estadístico de todos los algoritmos.

Para el parámetro del tamaño de la población, se cumple la normalidad de los residuos para todos los experimentos (bien por el test Kolmogorov-Smirnov, bien por el test Shapiro-Wilk), así como la homogeneidad de las varianzas (test de Levene), por lo que se aplicó el test de ANOVA, que resultó dar un valor de significación (*Sig.*) menor a 0,1 (*p-value* = 0,000), por lo que se debe rechazar la hipótesis del test de que las medias sean iguales. Esto quiere decir que las medias de los resultados de los distintos experimentos para ajustar la población de PBIL presentan diferencias estadísticamente significativas, por lo que los experimentos pueden ser considerados estadísticamente relevantes. Esto mismo se cumple para el parámetro tasa de aprendizaje, por lo que se puede concluir que los experimentos para ajustar dicho parámetro son relevantes estadísticamente por la misma razón.

Sin embargo, los parámetros probabilidad de mutación y cantidad de mutación no cumplen la propiedad de normalidad de los residuos para alguno de los experimentos de ajuste realizados, por lo que hubo de aplicarse el test de Kruskal-Wallis para determinar la relevancia. El resultado de este test para ambos conjuntos de experimentos obtuvo el mismo que el mostrado en la Tabla AI.21, lo cual también probó que dichos experimentos presentaban diferencias estadísticamente significativas, respecto a la mediana y a la distribución del resultado obtenido, esta vez, y por tanto concluimos que todos los experimentos de ajuste para el algoritmo PBIL presentan validez estadística.

Análisis de los resultados del algoritmo ABC

El análisis estadístico desarrollado con los datos provenientes de los experimentos realizados con el algoritmo ABC es el mismo que se explicó para las anteriores metaheurísticas (Figura 4.3). Igual que en los anteriores casos, se comenzó con un estudio de la dispersión de las muestras obtenidas en cada experimento. Al igual que para los demás algoritmos, las muestras de las pruebas realizadas con el algoritmo ABC presentan una baja dispersión, por lo que los datos pueden considerarse fiables atendiendo a este parámetro.

A continuación se realizó un estudio con cada una de las series de experimentos para ajustar los diferentes parámetros. Como se ha explicado en las anteriores secciones, para probar la relevancia estadística de los datos, hay que hacer un estudio respecto a las diferencias significativas que presentan los resultados obtenidos en los diferentes experimentos. Según se explicó en la Figura 4.3 para poder aplicar ANOVA se deben cumplir una serie de condiciones, entre las que están la homogeneidad de las varianzas y la normalidad residual de las muestras. Para el caso concreto del tamaño de la colonia, no se cumple la homocedasticidad de las varianzas (que se daría si el *p-value* > 0,1 en el test de Levene), por lo que no es posible aplicar ANOVA. En su lugar, se aplicó el test no paramétrico de Kruskal-Wallis, el cual arrojó un valor de significación inferior a 0.1 (*p-value* = 0.000), por lo que se prueba que hay diferencias significativas entre los datos provenientes de los experimentos con el tamaño de la colonia del ABC. Exactamente el mismo hecho ocurrió para el resto de los parámetros del algoritmo. Tanto los resultados obtenidos para ajustar la probabilidad de mutación, como los obtenidos con el ajuste de la proporción de las abejas obreras/observadoras de la colonia, o los de la configuración de la abeja exploradora, presentaron *p-values* inferiores a 0,1 para el test de Levene, por lo que las varianzas de los datos obtenidos con todos los experimentos del algoritmo ABC no son homogéneas. Por tanto, para todas las muestras de todos los experimentos se realizó el test no paramétrico de Kruskal-Wallis, arrojando el resultado esperado. El valor de significación de los datos fue menor a 0,1 en todos los casos, por lo que los datos provenientes de todos los experimentos

deben ser considerados significativamente distintos estadísticamente, por lo que son válidos científicamente.

Análisis de los resultados del algoritmo SS

El análisis estadístico efectuado a los datos provenientes de los experimentos realizados con el algoritmo SS es el mismo que se explicó para las anteriores metaheurísticas (Figura 4.3). Igual que en los anteriores casos, se comenzó con un estudio de la dispersión de las muestras obtenidas en cada experimento. Al igual que para los demás algoritmos, las muestras de las pruebas realizadas con la búsqueda dispersa presentan paradójicamente una baja dispersión, por lo que los datos pueden considerarse fiables atendiendo a este parámetro.

A continuación se realizó un estudio con cada uno de los principales bloques de experimentos que se realizaron para ajustar los diferentes parámetros del algoritmo. Como se ha explicado en las secciones anteriores, para probar la relevancia estadística de los datos, hay que hacer un estudio respecto a las diferencias significativas que presentan los resultados obtenidos en los diferentes experimentos. Según se explicó en la Figura 4.3, para poder aplicar ANOVA se deben cumplir una serie de condiciones, entre las que están la homogeneidad de las varianzas y la normalidad residual de las muestras. Para el caso concreto del tamaño de la población, no se cumple la homocedasticidad de las varianzas (*p-value* = 0,013 < 0,1 en el test de Levene), por lo que no es posible aplicar ANOVA. En su lugar, se aplicó el test no paramétrico de Kruskal-Wallis (K-W), el cual arrojó un valor de significación inferior a 0.1 (*p-value* = 0.000), por lo que se prueba que hay diferencias significativas entre los datos provenientes de los experimentos con el tamaño de la población del algoritmo SS. Este mismo hecho ocurrió para los experimentos del parámetro tamaño del conjunto de referencia. En este caso tampoco se cumplió la homogeneidad de las varianzas de los resultados obtenidos según el test de Levene. Para el caso del tamaño del RefSet, también se realizó la prueba de K-W, que igual que en los casos anteriores, arrojó un *p-value* < 0,1, por lo que quedó comprobado estadísticamente que los resultados provenientes de los experimentos referidos

al ajuste del tamaño del conjunto de referencia del algoritmo SS presentaban diferencias estadísticas significativas.

En cuanto a los dos parámetros principales restantes (proporción del conjunto de referencia y método de selección de padres), la propiedad que no se cumplió fue la normalidad de los residuos de las muestras correspondientes a algunos de los experimentos realizados. Ni el test Kolmogorov-Smirnov ni el Shapiro-Wilk fueron satisfechos para todas las pruebas realizadas, por lo que fue posible aplicar el test de ANOVA. Por este motivo, hubo que aplicar en este caso también el test no paramétrico de Kruskal-Wallis, el cual arrojó también el resultado esperado. El valor de significación de los datos fue menor a 0,1 en todos los casos, por lo que los datos provenientes de todos los experimentos pueden ser considerados significativamente distintos en base a la mediana y a la distribución del coste obtenido, y por tanto, los experimentos pueden ser considerados estadísticamente relevantes, y por tanto, científicamente válidos.

Análisis de los resultados del algoritmo GA

Finalmente, para probar la validez de los resultados obtenidos experimentalmente con el algoritmo GA, se realizó el completo análisis estadístico que se llevó a cabo para los datos de las demás estrategias, el cual se resume brevemente en esta sección. La metodología de análisis fue descrita en la Figura 4.3 de esta tesis. Al igual que en los anteriores análisis realizados, se comenzó con un estudio de la dispersión de las muestras obtenidas en cada experimento. Como en los demás algoritmos, las muestras de las pruebas realizadas con el algoritmo genético presentan una baja dispersión, por lo que los datos pueden considerarse fiables atendiendo a este parámetro.

A continuación se realizó un estudio con cada uno de los bloques de experimentos principales que se realizaron para ajustar los diferentes parámetros. Como se ha explicado en las anteriores secciones, para probar la relevancia estadística de los datos, hay que hacer un estudio respecto a las diferencias significativas que presentan los resultados obtenidos en los diferentes experimentos. Según se explicó en la Figura 4.3, para poder aplicar ANOVA se deben cumplir una serie de condiciones, entre las que están la homogeneidad de

las varianzas y la normalidad residual de las muestras. Para el caso concreto del tamaño de la población, estas dos propiedades se cumplen, ya que todos los test utilizados para realizar esas comprobaciones tienen un valor de significación mayor a 0,1. El test de Levene arroja un *p-value* igual a 0,195, por lo que las varianzas son constantes, mientras que los test bien de Kolmogorov-Smirnov o Shapiro-Wilk presentan valores superiores a 0,1, por lo que los residuos de los datos sometidos a estudio siguen una distribución normal. Así pues, se dan las condiciones para aplicar el test de ANOVA, que obtiene un valor de significación igual a 0,084 < 0,1, por lo que se rechaza la hipótesis de que las medias sean iguales. Por tanto, los experimentos relativos al tamaño de la población generan datos significativamente distintos, y por tanto relevantes estadísticamente.

En el caso del método de selección de padres utilizado por el GA, los datos arrojados en los distintos experimentos no satisfacen el test de Levene, lo que quiere decir que las varianzas no son homogéneas. Por tanto, la significación estadística de los datos debe comprobarse con el test no paramétrico de Kruskal-Wallis, que obtiene un valor de significación menor a 0,1 (*p-value* = 0,026), por lo que los datos arrojados por los experimentos son relevantes estadísticamente porque sus distribuciones de coste y sus medianas difieren significativamente.

En cuanto al parámetro que determina el número de hijos o de descendientes por generación, se hicieron pruebas con 1, 2, 4, 6 y 8 hijos por generación. Los datos obtenidos en tales experimentos superaron el test de Levene (con un *p-value* = 0,783 > 0,1) y los test de normalidad residual, con un *p-value* > 0,1 en todos los casos, por lo que se aplicó el test de ANOVA. El resultado fue un *p-value* = 0,369, que es mayor a 0,1, por lo que no se puede considerar que las medias obtenidas en estos experimentos resulten significativamente distintas. Este hecho se comprueba en la Figura 6.11, donde se observa que los resultados obtenidos con todas las configuraciones probadas son prácticamente iguales, por lo que se puede concluir que este bloque de experimentos resulta poco relevante para el ajuste final del algoritmo.

Por último, se analizaron los resultados procedentes del ajuste paramétrico de la probabilidad de mutación del GA. El ajuste de este parámetro sí que resultó ser muy significativo para el funcionamiento global del algoritmo, por lo que el análisis estadístico viene a confirmar lo que se infiere con un análisis empírico de

los datos. En este caso, los resultados de los experimentos para el ajuste de la probabilidad de mutación satisficieron tanto la normalidad residual (con *p-values* superiores a 0,1), como la homogeneidad de las varianzas, ya que el test de Levene arrojó un valor de significación igual a 0,156 > 0,1. Por tanto, se aplicó el test de ANOVA para examinar la similitud de las medias de los experimentos realizados con este parámetro. El test de ANOVA generó un *p-value* igual a 0,000, por lo que se verifica formalmente que los datos sometidos a estudio presentan diferencias significativas, con lo que se puede concluir que son relevantes estadística y científicamente.

Apéndice II

Listado de publicaciones que sustentan la tesis doctoral

En este apéndice se listan las publicaciones que se han obtenido durante el periodo de investigación durante el cual se ha elaborado esta tesis doctoral. Dichas publicaciones prueban el interés y validez del trabajo realizado durante estos años, ya que la mayoría han sido publicadas en foros científicos de prestigio, donde los trabajos son revisados por reconocidos investigadores especializados. Además queremos señalar que aunque todos los trabajos han sido fruto del esfuerzo de varios investigadores, en la mayoría de ellos (más del 90%) el autor de esta tesis ha sido el responsable principal, por lo que aparece como primer firmante de los mismos.

Asimismo, para una mayor claridad, se ha realizado una clasificación de todos los trabajos atendiendo en primer lugar a su relación con la temática de la tesis, y a continuación atendiendo al tipo y calidad de la publicación. La tabla AII.1 y los gráficos de las Figuras AII.1 y AII.2 resumen las publicaciones conseguidas teniendo en cuenta dichas consideraciones.

Publicaciones relacionadas directamente con la temática de la tesis

Se han conseguido un total de 21 publicaciones relacionadas directamente con la temática de la tesis, esto es, con la resolución del problema de optimización con

el que se ha trabajado, el FAP, aplicando técnicas metaheurísticas que normalmente han sido combinadas con otros algoritmos heurísticos y/o con estrategias basadas en paralelismo. Entre todas las publicaciones conseguidas, destacan dos revistas con factor de impacto (hay un tercer artículo enviado a la revista JCR *Expert Systems with Applications,* que está pendiente de contestación), una revista internacional, dos capítulos de libro internacionales, cinco congresos internacionales publicados en la serie *Lecture Notes in Computer Science,* y cuatro congresos internacionales IEEE o ACM (ver Tabla AII.1). A continuación aparecen las referencias de todas ellas.

Tabla AII.1. Resumen de las publicaciones relacionadas directamente con la temática de la tesis doctoral

Revistas JCR	*Engineering Optimization*			*Soft Computing*		**2**
Revista Internacional	*International Journal of Reasoning-based Intelligence Systems*					**1**
Capítulos de libro internacionales	*Computer Search Algorithms*			*New Trends in Artificial Intelligence*		**2**
Congresos LNCS	*EvoCOMNET 2008*	*HAIS 2008*	*PARA 2008*	*GPC 2009*	*EUROCAST 2009*	**5**
Congresos IEEE o ACM	*PDP 2008*	*GECCO 2008*		*ADVCOMP 2008* (x2)		**4**
Otros congresos internacionales	*DCAI 2010 (Springer), IASTED 2008, …*					**4**
Congresos nacionales	*Jornadas de Paralelismo 2008 y 2009*					**3**
TOTAL	***21 publicaciones***					

Revistas con factor de impacto (JCR)

1. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Optimizing a realistic large-scale frequency assignment problem using a new parallel evolutionary approach. *Engineering Optimization*, Taylor & Francis, 1-33, 2011. F.I.: 0.966-2009 (Revista del segundo cuartil)

2. F. Luna, C. Estébanez, C. León, J. M. Chaves-González, A. J. Nebro, R. Aler, C. Segura, M. A. Vega-Rodríguez, E. Alba, J. M. Valls, G. Miranda, J. A. Gómez-Pulido. Optimization Algorithms for Large-Scale Real-World Instances of the Frequency Assignment Problem. *Soft Computing*, Springer, 15(5), 975-990, 2011. F.I.: 1.328-2009 (Revista del segundo cuartil)

Otras revistas internacionales

3. J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Tuning the PBIL Algorithm to Solve a Real-World FAP Problem. *International Journal of Reasoning-based Intelligent Systems*, Inderscience Publishers, 2(1), 2-12, 2010

Capítulos de libro internacionales

4. D. Domínguez-González, J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Using PBIL for Solving a Real-World Frequency Assignment Problem in GSM Networks. In *New Trends in Artificial Intelligence*, J. Neves, M.F. Santos, J.M. Machado (Eds), 207-218, 2007

5. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Studying different heuristic searches to solve a real-world frequency assignment problem. In *Computer Search Algorithms*, Nova Science Publishers, 1-16, 2010 (*En prensa. Aceptado en Agosto de 2010*)

Congresos internacionales publicados en LNCS

6. J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. SS vs PBIL to Solve a Real-World Frequency Assigment Problem in GSM Networks. In *Proceedings of the 2008 conference on Applications of Evolutionary Computing, EvoCOMNET 2008 (LNCS 4974),* 21-30, 2008

7. J. M. Chaves-González, M. da Silva-Maximiano, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Comparing Hybrid Versions of SS and DE to Solve a Realistic FAP Problem. In *Proceedings of the 3rd international workshop on Hybrid Artificial Intelligence Systems, HAIS 2008 (LNAI 5271)*, 257-264, 2008 (Congreso Core C)

8. J. M. Chaves-González, R. Hernando-Carnicero, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Solving a Realistic FAP Using GRASP and Grid Computing. In *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing, GPC 2009 (LNCS 5529)*, 79-90, 2009 (Congreso Core C)

9. J.M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Solving a Real–World FAP Using the Scatter Search Metaheuristic. In *12th International Conference on Computer Aided Systems Theory, EUROCAST 2009 (LNCS 5717)*, 785–792, 2009

10. J. M. Chaves-González, M. A. Vega-Rodríguez, R. Salinas-Escribano, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. PBIL and Grid Computing to Resolve a Realistic FAP. In *9th International Workshop on State-of-the-Art in Scientific and Parallel Computing, PARA 2008 (Revised Selected Papers LNCS)*, 1-10, 2010 *(En prensa)*

Congresos internacionales publicados por IEEE o ACM

11. J. M. Chaves-González, D. Domínguez-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Parallelizing PBIL for Solving a Real-World Frequency Assignment Problem in GSM Networks. In *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*, 391-398, 2008 (Congreso Core C)

12. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Studying Different Variants of PBIL to Solve a Real-World FAP Problem in GSM Networks. In *The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '08)*, 83–88, 2008

13. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Population Based Incremental Learning to Solve the FAP Problem. In *The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '08)*, 123–128, 2008

14. F. Luna, C. Estébanez, C. León, J. M. Chaves-González, E. Alba, R. Aler, C. Segura, M. A. Vega-Rodríguez, A. J. Nebro, J. M. Valls, G. Miranda, J. A. Gómez-Pulido. Metaheuristics for Solving a Real-World Frequency Assignment Problem in GSM Networks. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, 1579-1586, 2008 (Congreso Core A)

Otros congresos internacionales

15. J. M. Chaves-González, D. Domínguez-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Using Cluster Computing to Solve a Real-World FAP Problem. In *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, ACTA Press, 202-207, 2008

16. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Improving Scatter Search to Solve a Real-World FAP. In *The 12th International Conference on Computer Aided Systems–Book of Extended Abstracts*, A. Quesada, J. Rodriguez, R. Moreno. IUCTC, 271-273, 2009

17. J. M. Chaves-González, M. A. Vega-Rodríguez, R. Salinas-Escribano, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Grid computing for Solving a Real-World FAP problem. In *The 9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, 1-4, 2008

18. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Swarm Intelligence, Scatter Search and Genetic Algorithm to Tackle a Realistic Frequency Assignment Problem. In *The International Symposium on Distributed and Artificial Intelligence (DCAI 2010)*, 441-448, 2010

Congresos nacionales

19. J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Planificación de frecuencias en redes GSM utilizando PBIL y computación clúster. En *XIX Jornadas de Paralelismo*, 232-237, 2008

20. J. M. Chaves-González, M. A. Vega-Rodríguez, R. Salinas-Escribano, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Computación grid para resolver un problema real de planificación de frecuencias. En *XIX Jornadas de Paralelismo*, 151-156, 2008

21. J. M. Chaves-González, R. Hernando-Carnicero, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Resolución del problema de la planificación de frecuencias en redes GSM utilizando GRASP y computación grid. En *XX Jornadas de Paralelismo*, 135-140, 2009

Publicaciones no relacionadas directamente con la temática de la tesis

Por otro lado, se han conseguido otras publicaciones que no están relacionadas directamente con la temática de la tesis, pero que han sido fruto del trabajo de investigación llevado a cabo durante la formación del doctorando. Algunos de ellos son el resultado de trabajos realizados en los cursos de doctorado, mientras que otros lo son del trabajo llevado a cabo por el doctorando en una línea de investigación previa al problema de optimización que dio forma a esta tesis. En cualquier caso, todas las publicaciones contribuyeron de alguna manera a la formación como investigador del autor de esta tesis doctoral, por lo que se consideró conveniente mencionarlas en este apéndice.

En total, se han conseguido 13 publicaciones que no están relacionadas directamente con la temática de esta tesis, entre las que destacan dos publicaciones en revistas con factor de impacto y dos capítulos de libro internacionales. A continuación aparecen las referencias de todas ellas.

Revistas con factor de impacto (JCR)

22. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Detecting Skin in Face Recognition Systems: A Colour Spaces Study. *Digital Signal Processing*, Elsevier, 20(3), 806-823, 2010. F.I.: 1.317-2009 (Revista del segundo cuartil)

23. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Pipeline-scheduling Simulator for Educational Purpose. *Journal of Universal Computer Science (J.UCS)*, Springer, 13(7), 959-969, 2007. F.I.: 0.315 (Revista del cuarto cuartil)

Capítulos de libro internacionales

24. J. M. Chaves-González, N. Otero-Mateo, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Game Implementation: An Interesting Strategy to Teach Genetic Algorithms. In *Computers and Education: E-learning, From Theory to Practice*, Springer, 205-223, 2007

25. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Fast Face Detector and Recognition for Biometrical Security System. In *Computer Vision Research Process*, Nova Science Publishers, 167-183, 2007

Congresos internacionales

26. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. PipeSim: Pipeline-Scheduling Simulator. In *8th International Symposium on Computers in Education (SIIE'06)*, 109-116, 2006

27. J. M. Chaves-González, M. A. Vega-Rodríguez, P. Martínez-Cobo, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Diffuse Matrix: An Optimized Data Structure for the Storage and Processing of Hyperspectral Images. In *International Conference on Signal Processing and Multimedia Applications (Sigmap 2007)*, 39-44, 2007

28. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Colour Spaces Study for Skin Colour Detection in Face Recognition Systems. In *International Conference on Signal Processing and Multimedia Applications (Sigmap 2007)*, 175-178, 2007

Congresos nacionales

29. J. M. Chaves-González, N. Otero-Mateo, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Diseño de juegos como herramienta didáctica para el aprendizaje de algoritmos genéticos. En Quinto Congreso Español de *Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2007)*, 457-464, 2007

30. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. PipeSim: simulador para la planificación de unidades funcionales segmentadas. En *XII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2006)*, 511-518, 2006

31. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Simulador para la planificación de pipelines. En *XVII Jornadas de Paralelismo*, 531-536, 2006

32. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Sistema de seguridad biométrico basado en extracción geométrica de características faciales. En *II Simposio sobre Seguridad Informática*, 53-60, 2007

33. J. M. Chaves-González, M. A. Vega-Rodríguez, P. Martínez-Cobo, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Estructura de datos optimizada para el almacenamiento y tratamiento de imágenes hiperespectrales. En *XVII Congreso Español de Informática Gráfica (CEIG'07)*, 263-266, 2007

34. J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Detección de piel humana en sistemas de reconocimiento facial usando diferentes espacios de colores: un estudio cuantitativo. En *XVII Congreso Español de Informática Gráfica (CEIG'07)*, 255-258, 2007

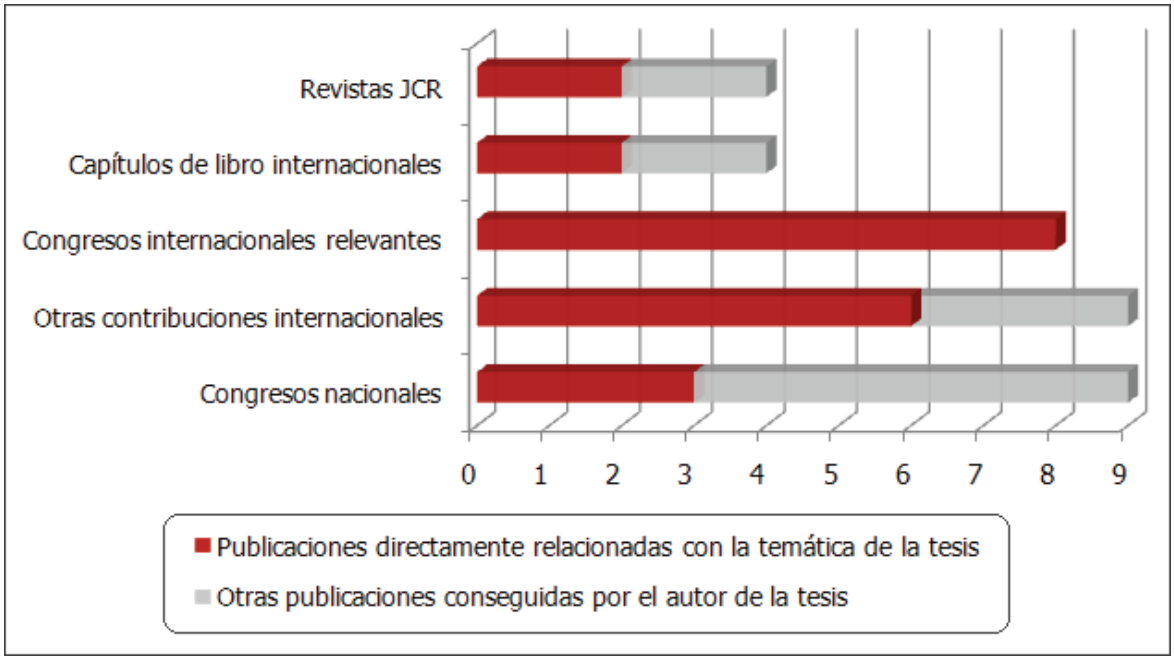

Figura AII.1. Resumen de todas las publicaciones conseguidas por el autor de la tesis durante su periodo predoctoral clasificadas según su relevancia científica

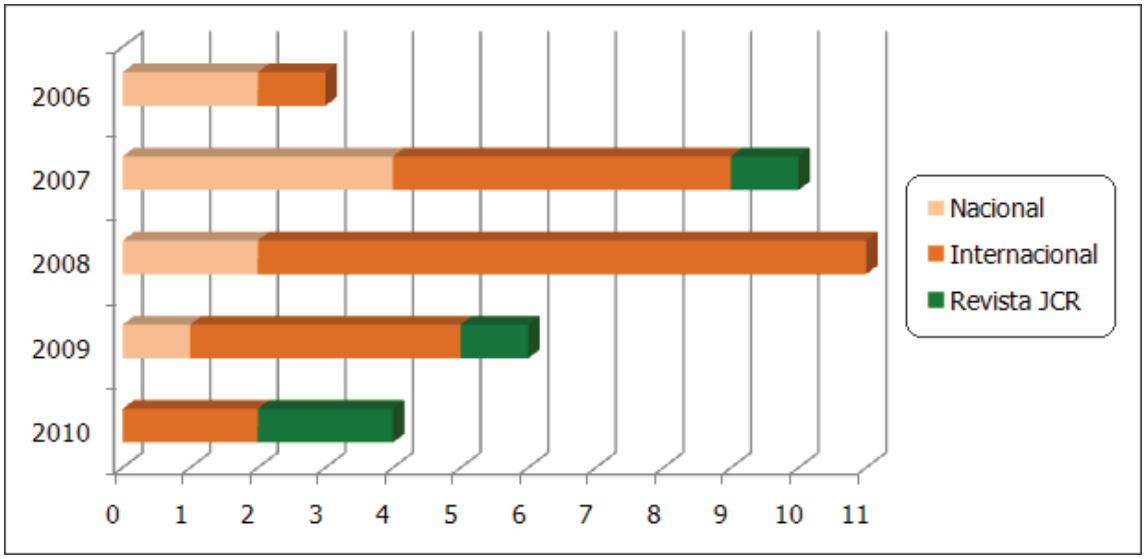

Figura AII.2. Resumen de las publicaciones conseguidas por el autor de la tesis durante su periodo predoctoral clasificadas de acuerdo al año que se obtuvieron y a su carácter (publicación nacional, internacional o revista JCR)

Referencias

[1] W.K. Hale. Frequency assignment: Theory and applications. Proceedings of the IEEE, 68 (12), 1497-1514, 1980

[2] M. Mouly and M. B. Paulet. The GSM System for Mobile Communications. Mouly et Paulet, Palaiseau, 1992

[3] GSM World statistics: http://www.gsmworld.com/newsroom/market-data/market_data_summary.htm, 2009

[4] J. Rapeli. UMTS: Targets, system concept, and standardization in a global framework. IEEE Personal Communications, 2 (1), 30-37, 1995

[5] A.H. Khan, M.A. Qadeer, J.A. Ansari, S. Waheed. 4G as a Next Generation Wireless Network. International Conference on Future Computer and Communication, 334-338, 2009

[6] S. Ahmadi. An overview of next-generation mobile WiMAX technology, IEEE Communications Magazine, 47 (6), 84-98, 2009

[7] K. I. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. Annals of Operations Research, 153 (1), 79-129, 2007

[8] FAP web: http://fap.zib.de/, 2010

[9] S. Kotrotsos, G. Kotsakis, P. Demestichas, E. Tzifa, V. Demesticha, and M. Anagnostou. Formulation and computationally efficient algorithms for an interference-oriented version of the frequency assignment problem. Wireless Personal Communications, 18, 289-317, 2001

[10] A. Eisenblätter. Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds. PhD thesis, Technische Universität Berlin, 2001

[11] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys, 35 (3), 268-308, 2003

[12] E.K. Burke and G. Kendall. Search methodologies: introductory tutorials in optimization and decision support techniques, Springer, 2005

[13] F.W. Glover and G. A. Kochenberger. Handbook of Metaheuristics (International Series in Operations Research & Management Science), Springer, 2003

[14] Web del proyecto OPLINK::UMA: http://oplink.lcc.uma.es, 2007

[15] B.H. Metzger. Spectrum management technique. 38th National ORSA Meeting, 1970

[16] FAP web bibliography: http://fap.zib.de/biblio/, 2010

[17] R. Borndörfer, A. Eisenblätter, M. Grötschel, A. Martin. Frequency assignment in cellular phone networks. Annals of Operations Research, 76, 73-93, 1998

[18] D. Brélaz. New methods to color the vertices of a graph. Communications of the ACM, 22, 251-256, 1979

[19] D. Costa. On the use of some known methods for t-colourings of graphs. Annals of Operations Research, 41, 343-358, 1993

[20] J.K. Hao, R. Dorne, P. Galinier. Tabu search for the frequency assignment in mobile radio networks. Journal of Heuristics, 4 (1), 47-62, 1998

[21] C. Valenzuela, S. Hurley, D.H. Smith. A permutation based genetic algorithm for minimum span frequency assignment. Parallel Problem Solving from Nature (PPSN V), LNCS 1498, 907-916, 1998

[22] A. Eisenblätter, M. Grötschel, A.M.C.A. Koster. Frequency planning and ramifications of coloring. Discussiones Mathematicae Graph Theory 22 (1), 51-88, 2002

[23] A.M.C.A. Koster. Frequency assignment - models and algorithms, Ph.D. thesis, Universiteit Maastricht, 1999 (Available at http://www.zib.de/koster/thesis.htm)

[24] R. Leese, S. Hurley. Methods and Algorithms for Radio Channel Assignment. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2002

[25] F. Luna. Metaheurísticas avanzadas para problemas reales en redes de telecomunicaciones. PhD thesis, Universidad de Málaga, 2008.

[26] M. Fischetti, C. Lepsch, G. Minerva, G. Romanin-Jacur. Toto, E. Frequency assignment in mobile radio systems using branch-and-cut techniques. European Journal of Operational Research, 123 (2), 241-255, 2000

[27] C. Mannino, A. Sassano. An enumerative algorithm for the frequency assignment problem. Discrete Applied Mathematics, 129, 155-169, 2003

[28] E-G. Talbi. A taxonomy of hybrid metaheuristics. Journal of Heuristics, 8 (2), 807-819, 2002

[29] J.H. Holland. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA, 1992 (first edition published in 1975)

[30] D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Longman Publishing, 1989

[31] T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Oxford University Press, 1997

[32] F. Glover, M. Laguna, and R. Martí. Scatter search. Advances in Evolutionary Computing: Theory and Applications, Springer, 519-537, 2003

[33] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. Control and Cybernetics, 29 (3), 653-684, 2000

[34] F. Glover. A template for Scatter Search and Path Relinking. In: J-K. Hao et al., editor, Artificial Evolution, LNCS 1363, 13-54, 1998

[35] R. Martí, M. Laguna, F. Glover. Principles of scatter search. European Journal of Operational Research, 169 (2), 359-372, 2006

[36] S. Baluja, R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. Twelfth Int. Conference on Machine Learning, San Mateo, CA, USA, 38-46, 1995

[37] S. Baluja. Population-Based Incremental Learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CS-94-163, Carnegie Mellon University, 1994

[38] P. Hansen, N. Mladenovic, J.A. Moreno-Pérez. Variable Neighbourhood Search. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 19 (2), 77-92, 2003

[39] N. Mladenovic and P. Hansen. Variable neighborhood search. Computers and Operations Research, 24 (11), 1097-1100, 1997

[40] H.R. Lourenço, O. Martin, and T. Stützle. Handbook of Metaheuristics, chapter Iterated local search, Kluwer Academic Publishers, 321-353, 2002

[41] T. Stützle. Local search algorithms for combinatorial problems analysis, algorithms and new applications. Technical report, DISKI Dissertationen zur Künstliken Intelligenz. Sankt Augustin, Germany, 1999

[42] T.A. Feo and M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, 6, 109-133, 1995

[43] M.G.C. Resende, C.C. Ribeiro. Greedy Randomized Adaptive Search Procedures. AT&T Labs Research Technical Report, 1-27, 2001

[44] D. Karaboga and B. Basturk. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. P. Melin et al. (Eds.), LNAI 4529, Springer, 789-798, 2007

[45] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, Springer, 39, 459-471, 2007

[46] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005

[47] E. Alba, editor. Parallel Metaheuristics: A New Class of Algorithms. Wiley, 2005

[48] E. Alba, F. Luna, and A. J. Nebro. Parallel heterogeneous genetic algorithms for continuous optimization. Parallel Computing, 30 (5-6), 669-719, 2004

[49] E. Alba, F. Luna, and A. J. Nebro. Advances in parallel heterogeneous genetic algorithms for continuous optimization. International Journal of Applied Mathematics and Computer Science, 14 (3), 117-133, 2004

[50] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 6 (5), 443-462, 2002

[51] S. Benkhider, A.R. Baba-Ali, H.A. Drias. A new generationless parallel evolutionary algorithm for combinatorial optimization. IEEE Congress on Evolutionary Computation, CEC 2007, 4691-4697, 2007

[52] T.G. Crainic and M. Toulouse. Parallel strategies for metaheuristics. In: F. W. Glover and G. A. Kochenberger, editors, Handbook of Metaheuristics, Norwell, MA, USA, Kluwer Academic Publishers, 2003

[53] V.-D. Cung, S.L. Martins, C.C. Ribeiro, and C. Roucairol. Strategies for the Parallel Implementation of Metaheuristics. In: C.C. Ribeiro and P. Hansen, editors, Essays and Surveys in Metaheuristics, Norwell, MA, USA, Kluwer Academic Publishers, 263-308, 2003

[54] D. Lim, Y-S. Ong, Y. Jin, B. Sendhoff, and B-S. Lee. Efficient hierarchical parallel genetic algorithms using grid computing. Future Generation Computer Systems, 23 (4), 658-670, 2007

[55] E. Alba, F. Chicano, S. Nesmachnow, H. Cancela, Parallel Metaheuristics in Telecommunications, E. Alba (editor) Parallel Metaheuristics. A New Class of Algorithms, chapter 20, Wiley, 495-516, 2005

[56] J. M. Chaves-González, R. Hernando-Carnicero, M. A. Vega-Rodríguez, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Solving a Realistic FAP Using GRASP and Grid Computing. Advances in Grid and Pervasive Computing (GPC 2009), Springer, 79-90, 2009

[57] J. M. Chaves-González, D. Domínguez-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, J. M. Sánchez-Pérez. Parallelizing PBIL for Solving a Real-World Frequency Assignment Problem in GSM Networks. 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008), 391-398, 2008

[58] J.M. Chaves-González, D. Domínguez-González, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, and M. Sánchez-Pérez. Using Cluster Computing to Solve a Real-World FAP Problem. Parallel and Distributed Computing and Networks (PDCN 2008), 202-207, 2008

[59] W. Crompton, S. Hurley, and N. M. Stephens. A parallel genetic algorithm for frequency assignment problems. Proceedings of the IMACS/IEEE Conference on Signal Processing, Robotics and Neural Networks, 81-84, 1994

[60] W. Crompton, S. Hurley, and N. M. Stephens. Frequency assignment using a parallel genetic algorithm. Proc. 2nd IEE/IEEE Workshop on Natural Algorithms in Signal Processing (NASP'93), 26/1-26/8, 1993

[61] N. Funabiki, Y. Takefuji. A neural network parallel algorithm for channel assignment problems in cellular radio networks. IEEE Transactions on Vehicular Technology, 41 (4), 430-437, 1992

[62] G. Kendall, M. Mohamad. Channel assignment optimisation using a hyper-heuristic. IEEE Conference on Cybernetics and Intelligent Systems, 2, 791-796, 2004

[63] G. Kendall, M. Mohamad. Channel Assignment in Cellular Communication Using a Great Deluge Hyper-Heuristic. IEEE International Conference on Network (ICON2004), 769-773, 2004

[64] H. Mabed, A. Caminada, J.K. Hao, D. Renaud. A dynamic traffic model for frequency assignment. Parallel Problem Solving from Nature (PPSN VII), LNCS 2439, 779-788, 2002

[65] C. Maple, L. Guo, and J. Zhang. Parallel genetic algorithms for third generation mobile network planning. International Conference on Parallel Computing in Electrical Engineering, 229-236, 2004

[66] E. Hart, P.Ross, J. Nelson, Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder, Evolutionary Computation, 6 (1), 61-80, 1998

[67] E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross & S. Schulenburg, Hyper-Heuristics: An Emerging Direction in Modern Search Technology, Handbook of Metaheuristics (eds. F.Glover & G.Kochenberger), Kluwer, 457-474, 2003

[68] K. Chakhlevitch and P.I. Cowling. Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristic Framework. 5th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP2005), LNCS 3448, 23-33, 2005

[69] E. Ozcan, B. Bilgin, E.E. Korkmaz. A Comprehensive Analysis of Hyper-heuristics, Intelligent Data Analysis, 12 (1), 3-23, 2008

[70] P. Ross, Hyper-heuristics, Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques (eds. E.K. Burke & G. Kendall), Springer, 529-556, 2005

[71] B. Bilgin, E. Ozcan, E.E. Korkmaz. An Experimental Study on Hyper-Heuristics and Final Exam Scheduling, Proceedings of the 2006 International Conference on the Practice and Theory of Automated Timetabling, 123-140, 2006

[72] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu. A graph-based hyper-heuristic for educational timetabling problems, European Journal of Operational Research, 176 (1), 177-192, 2007

[73] E.K. Burke, A. Meisels, S. Petrovic, R. Qu. A Graph-Based Hyper Heuristic for Timetabling Problems. European Journal of Operational Research, 176, 177-192, 2007

[74] E.K. Burke, S. Petrovic, and R. Qu. Case Based Heuristic Selection for Timetabling Problems, Journal of Scheduling, 9 (2), 1094-1136, 2006

[75] E.K. Burke, S. Petrovic, R. Qu. Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems, The Next Wave in Computing, Optimization, and Decision Technologies (eds. B.L. Golden, S. Raghavan & E.A. Wasil), Springer, 79-91, 2005

[76] E.K. Burke and J.P. Newall. Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings, Annals of Operations Research, 129, 107-134, 2004

[77] E.K. Burke, G. Kendall and E. Soubeiga. A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. Journal of Heuristics, 9 (6), 451-470, 2003

[78] E.K. Burke, B. MacCarthy, S. Petrovic, R. Qu. Knowledge Discovery in a Hyper-Heuristic Using Case-Based Reasoning on Course Timetabling. Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, Gent, Belgium, LNCS 2740, 276-286, 2002

[79] N.R. Sabar, M. Ayob. Examination timetabling using scatter search hyper-heuristic. 2nd Conference on Data Mining and Optimization (DMO '09), 127-131, 2009

[80] E. Ozcan, Y. Bykov, M. Birben, E.K. Burke. Examination timetabling using late acceptance hyper-heuristics. IEEE Congress on Evolutionary Computation (CEC '09), 997-1004, 2009

[81] A. Cuesta, L. Garrido, and H. Terashima. Building Hyper-heuristics through Ant Colony Optimization for the 2D Bin Packing Problem, KES'05, LNCS 3684, 654-660, 2005

[82] E.K. Burke, M. Hyde, G. Kendall, J. Woodward. A Genetic Programming Hyper-Heuristic Approach for Evolving 2-D Strip Packing Heuristics. IEEE Transactions on Evolutionary Computation, 1 (99), 1-1, 2010

[83] P. Ross, S. Schulenburg, J.G. Marín-Blázquez and E. Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), New York, 942-948, 2002

[84] J. de Armas, C. León, G. Miranda. Fine and Coarse-grained Parallel Algorithms for the 2D Cutting Stock Problem, 17th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2009), 255-262, 2009

[85] C. León, G. Miranda, C. Segura. A memetic algorithm and a parallel hyperheuristic island-based model for a 2D packing problem. 11th annual conference on Genetic and evolutionary computation (GECCO 2009), 1371-1378, 2009

[86] P. Cowling, G. Kendall, and E. Soubeiga, A Hyper-heuristic Approach to Scheduling a Sales Summit, LNCS 2079, PATAT III, Konstanz, Germany, selected papers (eds E.K. Burke and W. Erben), 176-190, 2000

[87] G. Ochoa, J.A. Vazquez-Rodriguez, S. Petrovic, E. Burke. Dispatching rules for production scheduling: A hyper-heuristic landscape analysis. IEEE Congress on Evolutionary Computation (CEC '09), 1873-1880, 2009

[88] R. Bai, E.K. Burke, and G. Kendall. Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. Journal of the Operational Research Society, 59 (10), 1387-1397, 2008

[89] R. Bai, G. Kendall. An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-heuristics, In: T. Ibaraki, K. Nonobe, M. Yagiura (eds.), Meta-heuristics: Progress as Real Problem Solvers, Selected Papers from the 5th Metaheuristics International Conference (MIC 2003), Springer, 87-108, 2005

[90] E.K. Burke, J.D. Landa Silva, E. Soubeiga. Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling. In: T. Ibaraki, K. Nonobe, M. Yagiura (eds.), Meta-heuristics: Progress as Real Problem Solvers, Springer, 2005

[91] U. Aickelin, E.K. Burke and J. Li. An Estimation of Distribution Algorithm with Intelligent Local Search for Rule-based Nurse Rostering. Journal of the Operational Research Society, Palgrave Macmillan, 2006

[92] R. Bai, E.K. Burke, G. Kendall, J. Li, B. McCollum. A Hybrid Evolutionary Approach to the Nurse Rostering Problem. IEEE Transactions on Evolutionary Computation, 14 (4), 580-590, 2010

[93] P. Cowling, K. Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel, Proceedings of the Congress on Evolutionary Computation, 2, 1214-1221, 2003

[94] P. Cowling, G. Kendall, & L. Han. An Adaptive Length Chromosome Hyperheuristic Genetic Algorithm for a Trainer Scheduling Problem. Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02), Singapore, 267-271, 2002

[95] S.M. Remde, P.I. Cowling, K.P. Dahal, and N. Colledge. Exact/Heuristic Hybrids using rVNS and Hyperheuristics for Workforce Scheduling. European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP2007), LNCS 4446, 188-197, 2007

[96] A. Keles, A. Yayml, A.S. Uyar. Ant based Hyper Heuristic for Physical Impairment aware routing and wavelength assignment. IEEE Sarnoff Symposium, 1-5, 2010

[97] M. Ayob, G. Kendall. A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing For Multi Head Placement Machine. Proceedings of the International Conference on Intelligent Technologies (InTech'03), Thailand, 132-141, 2003

[98] K.A. Dowsland, E. Soubeiga, and E.K. Burke. A simulated annealing hyper-heuristic for determining shipper sizes. European Journal of Operational Research, 179 (3), 759-774, 2007

[99] C. Leon, G. Miranda, C. Segura. Optimizing the Configuration of a Broadcast Protocol through Parallel Cooperation of Multi-objective Evolutionary Algorithms. The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP '08), 135-140, 2008

[100] C. Segura, Y. González, G. Miranda, C. León. Parallel Hyperheuristics for the Antenna Positioning Problem. II International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2010), Springer, 469-476, 2010

[101] C. León, G. Miranda, C. Segura. Parallel hyperheuristic: a self-adaptive island-based model for multi-objective optimization. 10th annual conference on Genetic and evolutionary computation (GECCO 2008), 757-758, 2008

[102] C. León, G. Miranda, E. Segredo, C. Segura. Parallel Hypervolume-Guided Hyperheuristic for Adapting the Multi-objective Evolutionary Island Model. Studies in Computational Intelligence, 236, 261-272, 2009

[103] C. León, G. Miranda, C. Segura. Hyperheuristics for a Dynamic-Mapped Multi-Objective Island-Based Model. Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, LNCS 5518, 41-49, 2009

[104] L. Han, G. Kendall. Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. Proceedings of Congress on Evolutionary Computation (CEC2003), Canberra, Australia, 3, 2230-2237, 2003

[105] A. Acan, H. Altincay, Y. Tekol, A. Unveren A Genetic Algorithm with Multiple Crossover Operators for Optimal Frequency Assignment Problem. The 2003 Congress on Evolutionary Computation (CEC '03), 1, 256-263, 2003

[106] C. Crisan and H. Mühlenbein. The breeder genetic algorithm for frequency assignment. Parallel Problem Solving from Nature (PPSN V), 897-906, 1998

[107] M. Cuppini. A genetic algorithm for channel assignment problems, European Transactions on telecommunications and related technologies, 5, 285-294, 1994

[108] R. Dorne and J.-K. Hao. An evolutionary approach for frequency assignment in cellular radio networks. Proc. of the IEEE Int. Conf. on Evolutionary Computation, 539-544, 1995

[109] L. Idoumghar and R. Schott. Two Distributed Algorithms for the Frequency Assignment Problem in the Field of Radio Broadcasting. IEEE Transactions on Broadcasting, 55 (2), 223-229, 2009

[110] L. Idoumghar, R. Schott. A New Hybrid GA-MDP Algorithm For The Frequency Assignment Problem. 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '06), 18-25, 2006

[111] L.M. San Jose-Revuelta, A new adaptive genetic algorithm for fixed channel assignment. Information Sciences, 177 (13), 2655-2678, 2007

[112] M. Alabau, L. Idoumghar, R. Schott. New hybrid genetic algorithms for the frequency assignment problem. IEEE Transactions on Broadcasting, 48 (1), 27-34, 2002

[113] M. Alabau, L. Idoumghar, R. Schott. New hybrid genetic algorithms for the frequency assignment problem. Proceedings of the 13th International Conference on Tools with Artificial Intelligence, 136-142, 2001

[114] S. Matsui, I. Watanabe, K.I. Tokoro. An efficient hybrid genetic algorithm for a fixed channel assignment problem with limited bandwidth. Genetic and Evolutionary Computation Conference (GECCO 2003), LNCS 2724, 2240-2251, 2003

[115] J.Y. Greff, L. Idoumghar, R. Schott. Application of markov decision processes to the frequency assignment problem. Applied Artificial Intelligence, 18 (8), 761-773, 2004

[116] S. Salcedo-Sanz, C. Bousoño-Calzón. A portable and scalable algorithm for a class of constrained combinatorial optimization problems. Computers & Operations Research, 32, 2671-2687, 2005

[117] G. Colombo. A genetic algorithm for frequency assignment with problem decomposition. International Journal of Mobile Network Design and Innovation, 1 (2), 102-112, 2006

[118] F. Luna, E. Alba, A.J. Nebro, S. Pedraza. Evolutionary algorithms for real-world instances of the automatic frequency planning problem in GSM networks. Seventh European Conference on Evolutionary Computation in Combinatorial Optimization (EVOCOP 2007), LNCS 4446, 108-120, 2007

[119] F. Luna, C. Blum, E. Alba, A.J. Nebro. ACO vs EAs for Solving a Real-World Frequency Assignment Problem in GSM Networks. GECCO'07, London, UK, 94-101, 2007

[120] F. Luna, C. Estébanez, C. León, J.M. Chaves-González, E. Alba, R. Aler, C. Segura, M.A. Vega-Rodríguez, A.J. Nebro, J.M. Valls, G. Miranda, J.A. Gómez-Pulido. Metaheuristics for Solving a Real-World Frequency Assignment Problem in GSM Networks. Genetic and Evolutionary Computation Conference (GECCO 2008), 1579-1586, 2008

[121] J.M. Chaves-González, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, J.M. Sánchez-Pérez. Swarm Intelligence, Scatter Search and Genetic Algorithm to Tackle a Realistic Frequency Assignment Problem. The International Symposium on Distributed and Artificial Intelligence (DCAI 2010), 441-448, 2010

[122] F. Luna, C. Estébanez, C. León, J.M. Chaves-González, A.J. Nebro, R. Aler, C. Segura, M.A. Vega-Rodríguez, E. Alba, J.M. Valls, G. Miranda, J.A. Gómez-Pulido. Optimization Algorithms for Large-Scale Real-World Instances of the Frequency Assignment Problem. Soft Computing, 15 (5), 975-990, 2011

[123] E-G. Talbi. Metaheuristics: from design to implementation. John Wiley and Sons, United Kingdom, 2009

[124] H.P. Hamiez, J.K. Hao. Scatter search for graph coloring. Artificial Evolution. LNCS 2310, 195-213, 2002

[125] J. M. Chaves-González, M. A. Vega-Rodríguez, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Solving a Real–World FAP Using the Scatter Search Metaheuristic. 12th International Conference on Computer Aided Systems Theory, EUROCAST 2009 (LNCS 5717), 785–792, 2009

[126] J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. SS vs PBIL to Solve a Real-World Frequency Assignment Problem in GSM Networks. Applications of Evolutionary Computing, (EvoWorkshops 2008), LNCS 4974, 21-30, 2008

[127] J. M. Chaves-González, M. da Silva-Maximiano, M. A. Vega-Rodríguez, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Comparing Hybrid Versions of SS and DE to Solve a Realistic FAP Problem. Hybrid Artificial Intelligence System (HAIS 2008), LNAI 5271, 257-264, 2008

[128] J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Tuning the PBIL algorithm to solve a real-world FAP problem. International Journal of Reasoning-based Intelligent System (IJRIS), Inderscience Publishers, 43-52, 2009

[129] J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Population-Based Incremental Learning to solve the FAP problem. The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2008), 123-128, 2008

[130] J. M. Chaves-González, M. A. Vega-Rodríguez, D. Domínguez-González, J. A. Gómez-Pulido and J. M. Sánchez-Pérez. Studying different variants of PBIL to solve a real-world FAP problem in GSM networks. The Second International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2008), 83-88, 2008

[131] D. Domínguez-González, J.M. Chaves-González, M.A. Vega-Rodríguez, J.A. Gómez-Pulido, J.M. Sánchez-Pérez. Using PBIL for Solving a Real-World Frequency Assignment Problem in GSM Networks. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007, LNCS (LNAI) 4874, 207-218, 2007

[132] M. Dorigo, T. Stützle. Ant Colony Optimization. MIT Press, Cambridge, MA, 2004

[133] V. Maniezzo, A. Carbonaro. An ANT Heuristic for the Frequency Assignment Problem. Future Generation Computer Systems, 16, 927-935, 1999

[134] F.C. Gomes, P. Pardalos, C.S. Oliveira, M.G.C. Resende. Reactive GRASP with path relinking for channel assignment in mobile phone networks. Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM'01), 60-67, 2001

[135] X. Liu, P.M. Pardalos, S. Rajasekaran, and M.G.C. Resende. A GRASP for frequency assignment in mobile radio networks. Dimacs series on discrete mathematics and theoretical computer science, 52, 195-201, 2000

[136] C.E.C. Vieira, P.R.L. Gondim, C.A. Rodrigues, J.L. Bordim. A new technique to the channel assignment problem in mobile communication networks. IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2008), 1-5, 2008

[137] M. da Silva Maximiano, M.A. Vega-Rodriguez, J.A. Gomez-Pulido, J.M. Sanchez-Perez. Multiobjective frequency assignment problem using the MO-VNS and MO-SVNS algorithms. The World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), 221-226, 2009

[138] M. Chiarandini and T. Stützle. An application of Iterated Local Search to Graph Coloring Problem. Proceedings of the Computational Symposium on Graph Coloring and its Generalizations, 112-125, 2002

[139] P. Galinier, A. Hertz. A survey of local search methods for graph coloring. Computers and Operations Research, 33 (9), 2547-2562, 2006

[140] MPI Forum home page, http://www.mpi-forum.org/index.html, 2010

[141] F. Luna, A. J. Nebro, E. Alba, J. J. Durillo. Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm. Engineering Optimization, 40 (11), 1067-1084, 2008

[142] F. Luna, A. J. Nebro, J. J. Durillo, E. Alba. Large-scale real-world automatic frequency planning in GSM networks using GrEA. International Conference on Metaheuristics and Nature Inspired Computing (META 2008), Hammamet, Tunisia, 2008

[143] A.R. Mishra. Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G. Wiley, 2004

[144] J. M. Picard, Z. Altman, S. Ben Jamaa, M. Demars, H. Dubreil, B. Fourestié, and A. Ortega. Automatic cell planing strategies for umts networks. International Journal of Mobile Network Design and Innovation, 1 (1), 8–17, 2005

[145] Instancias benchmark del FAP: http://fap.zib.de/problems/, 2005

[146] S.A.G. Shirazi, H. Amindavar. Fixed Channel Assignment Using New Dynamic Programming Approach in Cellular Radio Networks. Computers and Electrical Engineering, 31 (4-5), 303-333, 2005

[147] B. H. Walke. Mobile Radio Networks: Networking, protocols and traffic performance. Wiley, 2002

[148] A.M.J. Kuurne. On GSM mobile measurement based interference matrix generation. IEEE 55th Vehicular Technology Conference, 1965-1969, 2002

[149] M.K. Simon and M-S. Alouini. Digital Communication over Fading Channels: A Unified Approach to Performance Analysis. Wiley, 2005

[150] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. Numerical Recipes in C++: The Art of Scientific Computing. Second edition, Cambridge University Press, 2002

[151] G.L. Nemhauser, L.A. Wolsey. Integer and Combinatorial Optimization. John Wiley & Sons, New York, 1988

[152] S. Priem Mendes. Heuristics Effectiveness Analysis for the Radio Network Design Problem. PhD thesis, Universidad de Extremadura, 2009

[153] G.A. Groes. A Method for Solving Traveling Salesman Problems. Operations Research, 6, 791-812, 1958

[154] S. Lin. Computer Solutions for the Traveling Salesman Problem. Bell Systems Technology Journal, 44, 2245-2269, 1965

[155] D.S. Johnson, C.H. Papadimitiou, M. Yannakakis. How Easy is Local Search?. Journal of Computer Science, 37, 79-100, 1988

[156] F. Glover. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13, 533-549, 1986

[157] T. Stützle. Local Search Algorithms for Combinatorial Problems-Analysis, Improvements and New Applications. PhD Thesis, Darmstadt, University of Technology, Department of Computer Science, 1998

[158] F. Glover. Heuristics for integer programming using surrogate constraints. Decision Sciences, 8, 156-166, 1977

[159] F. Glover, M. Laguna. Tabu Search, Kluwer Academic Publishers, 1997

[160] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21, 1087-1092, 1953

[161] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. Science, 220, 671-680, 1983

[162] V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization, Theory and Application, 45, 41-55, 1985

[163] P. Larrañaga, J.A. Lozano. Estimation of distribution algorithms. A new tool for evolutionary computation. Kluwer Academic Publishers, 2001

[164] A.S. Fraser. Generation of genetic systems by automatic digital computer. Australian Journal of Biological Science, 10, 484-491, 1957

[165] G.E.P. Box. Evolutionary operation: a method of increasing industrial productivity. Applied Statistics, 6, 81-101, 1957

[166] R. Friedberg. A learning machine, part I. IBM Journal of Research and Development, 2, 2-13, 1958

[167] L. Fogel, A. Owens, M. Walsh. Artificial intelligence through a simulation of evolution. Biophysics and Cybernetic Systems: Proceedings of the 2nd Cibernetic Sciences Symposium, 131-155, 1965

[168] I. Rechenberg. Evolutionsstrategie - el der de Prinzipien del nach de Systeme del technischer de Optimierung biologischen la evolución. PhD thesis, 1971 (reimpreso por Fromman-Holzboog en 1973)

[169] J.R. Koza. Genetic Programming: on the programming of computers by means of natural selection. MIT Press, 1992

[170] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD thesis, Dipartamento di Elettronica, Politecnico di Milano, 1992

[171] M. Dorigo, T. Stützle. Handbook of Metaheuristics, Kluwer Academic Publisher, 57, chapter: The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, 251-285, 2003

[172] J. Kennedy, R.C. Eberhart. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1942-1948, 1995

[173] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999), 1931-1938, 1999

[174] R. Storn, K. Price. Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. ICSI Technical Report TR-95-012, 1995

[175] K. Price, R.M. Storn, J.A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization, Springer, 2005

[176] J. Demsar. Statistical comparison of classifiers over multiple data sets. Journal of Machine Learning Research, 7, 1-30, 2006

[177] D. J. Sheskin. Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2003

[178] W. Navidi. Estadística para Ingenieros y Científicos, McGraw-Hill, 2006

[179] K. Hwang. Advanced computer architecture: Parallelism, scalability, programmability, McGraw-Hill, 1993

[180] H.S. Stone. High-Performance Computer Architecture (3rd Edition), Addison-Wesley, 1993

[181] Web oficial de OpenMP: www.openmp.org, 2010

[182] L. Smarr, C. Catlett. Metacomputing. Communications of the ACM, 35 (6), 44-52, 1992

[183] I. Foster. What is the grid? a three point checklist. Grid Today, 1 (6), 2002

[184] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputing Applications, 15 (3), 2002

[185] Web de Globus: www.globus.org/, 2010

[186] Web de GridWay: www.gridway.org, 2010

[187] Web de gLite: glite.web.cern.ch/glite/, 2010

[188] WMS project page: web.infn.it/gLiteWMS/, 2010

[189] Condor project homepage: www.cs.wisc.edu/condor/, 2010

[190] Web de ProActive: proactive.inria.fr/, 2010

[191] Web del proyecto OPLINK::ULL: http://www.oplink.ull.es/, 2007

[192] Scientific Linux Home Page: https://www.scientificlinux.org/, 2010

[193] MPICH2 Home Page: www.mcs.anl.gov/research/projects/mpich2/, 2010

[194] GCC Compiler Home Page: http://gcc.gnu.org/, 2010

[195] Web del Ceta-Ciemat: http://www.ceta-ciemat.es/, 2010

[196] Web del Proyecto EELA: http://www.eu-eela.org/first-phase.php, 2008

[197] J.M. Chaves-González, M.A. Vega-Rodríguez, D. Domínguez-González, J.A. Gómez-Pulido, J.M. Sánchez-Pérez. Planificación de frecuencias en redes GSM utilizando PBIL y computación clúster. XIX Jornadas de Paralelismo, 232-237, 2008

[198] X-S. Yang. Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-inspired Computation, 2 (2), 78-84, 2010

[199] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi. GSA: A Gravitational Search Algorithm. Information Sciences, 179 (13), 2232-2248, 2009

[200] IBM SPSS Statistics home page: http://www.spss.com/es/, 2010