UNIVERSIDAD
DE MÁLAGA

LENGUAJES Y
CIENCIAS DE LA
COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

TESIS DOCTORAL

Metaheurísticas avanzadas para problemas reales en redes de telecomunicaciones

Autor
Francisco Luna Valero

Directores
Dr. Enrique Alba Torres y Dr. Antonio J. Nebro Urbaneja

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

Abril de 2008

Los Drs. **Enrique Alba Torres** y **Antonio J. Nebro Urbaneja**, Titulares de Universidad del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga,

Certifican

que D. **Francisco Luna Valero**, Ingeniero en Informática por la Universidad de Málaga, ha realizado en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, bajo su dirección, el trabajo de investigación correspondiente a su Tesis Doctoral titulada

Metaheurísticas avanzadas para problemas reales en redes de telecomunicaciones

Revisado el presente trabajo, estimo que puede ser presentado al tribunal que ha de juzgarlo, y autorizo la presentación de esta Tesis Doctoral en la Universidad de Málaga.

En Málaga, Abril de 2008

Firmado: Dr. Enrique Alba Torres y Dr. Antonio J. Nebro Urbaneja
Titulares de Universidad
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga

Agradecimientos

Esta tesis doctoral existe, en parte, gracias al apoyo de muchas personas e instituciones que han contribuido a su realización durante estos años. En primer lugar, mi agradecimiento a Antonio y a Enrique, mis directores, por su ayuda, paciencia y dedicación. También a mis compañeros de laboratorio, que siempre han estado ahí cuando ha sido necesario.

Un agradecimiento muy especial a Ana, por su incondicional apoyo y su compresión durante las "ausencias" que ha supuesto este trabajo.

Desde luego esta tesis no hubiese existido si no es por el respaldo y apoyo de mi familia, que desde el primer momento ha confiado en mí y ha facilitado esta tarea todo lo posible.

Finalmente, me gustaría dedicar un especial reconocimiento al Ministerio de Educación y Ciencia y al Fondo Europeo de Desarrollo Regional mediante el contrato TIN2005-08818-C04 por la concesión de una beca FPI con la que ha sido posible andar este camino.

Índice general

Capítulo 1

Introducción

1.1. Planteamiento

El diseño de algoritmos cada vez más eficientes para resolver problemas complejos, tanto de optimización como de búsqueda, ha sido tradicionalmente uno de los aspectos más importantes de la investigación en Informática. El objetivo perseguido en este campo es, fundamentalmente, el desarrollo de nuevos métodos capaces de resolver los problemas complejos mencionados con el menor esfuerzo computacional posible, mejorando así los resultados obtenidos por los algoritmos existentes. En consecuencia, esto no sólo permite afrontar problemas actuales de forma más eficiente, sino también tareas vedadas en el pasado debido a su alto coste computacional. En este contexto, la actividad investigadora tanto en algoritmos exactos como en heurísticos *ad hoc* y metaheurísticos para resolver problemas complejos de optimización está creciendo de forma evidente en estos días. La razón es que continuamente se están afrontando nuevos problemas de ingeniería, mientras que, al mismo tiempo, cada vez se dispone de mejores recursos computacionales, como nuevos tipos de ordenadores, redes, y entornos como Internet.

La principal ventaja de la utilización de algoritmos exactos es que garantizan encontrar el óptimo global de cualquier problema, pero tienen el grave inconveniente de que en problemas reales (que suelen ser NP-duros en la mayoría de los casos) su tiempo de ejecución y/o los requisitos de memoria crecen de forma exponencial con el tamaño del problema. En cambio, los algoritmos heurísticos *ad hoc* son normalmente bastante rápidos, pero la calidad de las soluciones encontradas está habitualmente lejos de ser óptima. Otro inconveniente de estos heurísticos *ad hoc* es que no son fáciles de definir en determinados problemas. Las metaheurísticas[1] ofrecen un equilibrio adecuado entre ambos extremos: son métodos genéricos que ofrecen soluciones de buena calidad (el óptimo global en muchos casos) en un tiempo razonable.

La industria de las telecomunicaciones ha proporcionado, y sigue proporcionando, una gran cantidad de problemas de optimización que surgen desde el propio diseño del sistema de comunicación hasta algunos aspectos de su funcionamiento. La resolución de estos problemas ha jugado, sin lugar a dudas, un papel muy destacado en el desarrollo y utilización de este tipo de sistemas. No obstante, a medida que se han ido haciendo más populares y su penetración en el mercado es mayor, el tamaño de los sistemas de telecomunicaciones ha ido creciendo y, por tanto, los problemas que plantean tienen una dimensión tan elevada que los hacen inabordables con técnicas exactas. Los algoritmos metaheurísticos son una de las mejores opciones en este contexto, ya que son capaces de encontrar soluciones de calidad en tiempos aceptables.

[1] Algunos autores consideran los algoritmos metaheurísticos una subclase de los algoritmos heurísticos. Por ese motivo usamos el término "heurístico *ad hoc*" para referirnos a los algoritmos aproximados pensados específicamente para un problema concreto.

Éste es el contexto donde se enmarca esta tesis doctoral. Nos proponemos aplicar técnicas metaheurísticas a problemas de optimización reales surgidos en la industria de las telecomunicaciones, analizando distintas posibilidades para sacar el máximo partido a dichas técnicas y ofrecer así soluciones de gran calidad. En concreto, los problemas abordados son la planificación de celdas, la asignación automática de frecuencias y la optimización del proceso de difusión en redes *ad hoc* de dispositivos móviles. Debido a las características de estos problemas, se han desarrollado metaheurísticas avanzadas que abarcan dos ámbitos: optimización multiobjetivo y sistemas de computación grid. El primer caso está motivado por la naturaleza multiobjetivo de dos de los problemas abordados (planificación de celdas y difusión óptima), en los que existen varias funciones contrapuestas que se han de optimizar a la vez. El segundo caso viene dado por la complejidad y dimensión de los problemas planteados, por lo que se han desarrollado modelos que permiten su ejecución en sistemas de computación grid, lo que nos va a permitir, no sólo reducir el tiempo de ejecución de los algoritmos, sino también obtener modelos de búsqueda que permitan resolver estos problemas de forma más efectiva.

1.2. Objetivos y fases

Los objetivos principales de esta tesis doctoral son: aplicar técnicas metaheurísticas a problemas de optimización reales en redes de telecomunicaciones, analizar los resultados para comprender el comportamiento de estos algoritmos y proponer nuevos métodos para resolver los problemas de manera más eficaz y eficiente. Hemos seguido las fases típicas establecidas por el método científico [130, 252], según definió F. Bacon:

1. **Observación**. Es el paso inicial de toda investigación. Observar es fijar cuidadosamente la atención en un hecho cualquiera para estudiarlo tal y como se presenta en la realidad. En nuestro caso, hemos analizado atentamente algunos de los problemas de optimización que surgen en el campo de las telecomunicaciones así como las técnicas que se han utilizado para resolver estos problemas. A partir de este análisis, hemos identificado las principales carencias de estos planteamientos.

2. **Hipótesis**. Tratando de resolver las carencias identificadas en el paso anterior, nuestra hipótesis consiste en proponer un conjunto de nuevas técnicas que permitan resolver estas carencias.

3. **Experimentación**. La experimentación en nuestro campo consiste en implementar las nuevas propuestas algorítmicas planteadas como hipótesis de trabajo así como la realización las pruebas correspondientes.

4. **Demostración o refutación de la hipótesis**. Se analizarán los resultados obtenidos con los nuevos modelos desarrollados, comparándolos exhaustivamente técnicas del estado del arte.

5. **Conclusiones**. Se presentarán las conclusiones a las que llegamos tras nuestro trabajo de investigación, y se sugerirán líneas de trabajo futuro que surgen de nuestro estudio.

El método científico, además, se sustenta en dos ejes básicos: la reproducibilidad y falsabilidad. Mientras el primero hace referencia a la condición de poder reproducir los efectos de un proceso siempre que se mantengan los condicionantes del mismo, el segundo establece que toda proposición científica tiene que ser susceptible de ser falsada. Respecto a la reproducibilidad, se presentan en todo momento los detalles necesarios para que los experimentos que se incluyen en este trabajo puedan reproducirse por cualquier otro investigador interesado. En cuanto a la falsabilidad, los

resultados de nuestros estudios se muestran de forma clara, estructurada y sencilla. Así, debido a la naturaleza estocástica de los algoritmos utilizados, se han realizado un mínimo de 30 pruebas independientes de cada experimento, además de un estudio que da validez estadística a las conclusiones obtenidas.

Las fases anteriores se han concretado en el esquema de trabajo que aparece en la Figura 1.1. Hemos estudiado tres problemas del campo de las telecomunicaciones, los dos primeros proceden de las redes de telefonía móvil, la planificación de celdas y la asignación de frecuencias, y, el tercero, se encuadra dentro de las redes ad hoc de dispositivos móviles: la optimización del proceso de difusión (*broadcasting*). En el siguiente paso se ha analizado las técnicas metaheurísticas, donde hemos propuesto dos técnicas con las que vamos a abordar los tres problemas, algoritmos evolutivos y búsqueda dispersa. Puesto que los problemas de la planificación de celdas y la optimización del proceso de difusión son multiobjetivo, hemos extendido estos algoritmos para trabajar con este tipo de tareas de optimización. Junto con los algoritmos evolutivos y la búsqueda dispersa, hemos utilizado un algoritmo específico para cada uno de los tres problemas estudiados (estrategias evolutivas, colonias de hormigas y algoritmos genéticos celulares, respectivamente). Nos hemos planteado, por tanto, tres metaheurísticas para la resolución de cada problema. Finalmente, dado el elevado coste computacional asociado a estos problemas de corte y dimensiones reales, también hemos propuesto una extensión de las técnicas de optimización de forma que se puedan ejecutar en sistemas de computación grid. Una vez estudiados los problemas y propuestas las técnicas, se aplican los algoritmos y se analizan los resultados obtenidos.

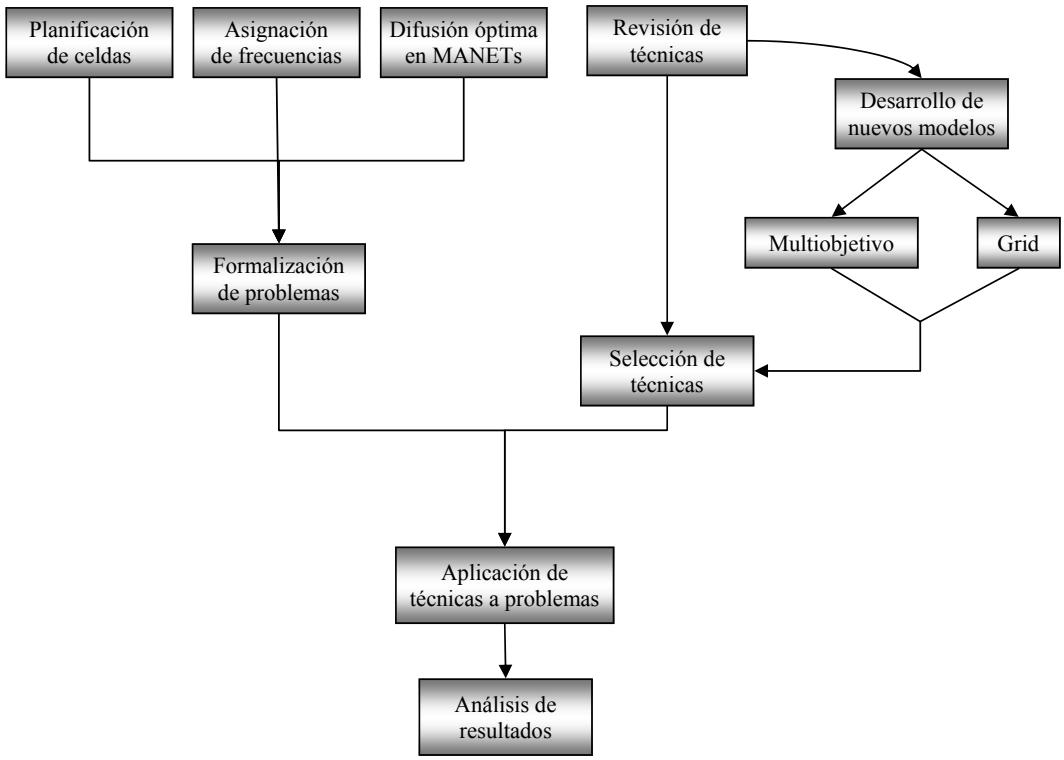


Figura 1.1: Fases seguidas durante la elaboración de esta tesis.

1.3. Contribuciones

En este apartado se listan las contribuciones de la presente tesis. De forma esquemática, estas contribuciones se pueden resumir como sigue:

- Desarrollo de un nuevo modelo matemático para el problema de la asignación de frecuencias donde se tiene en cuenta información muy precisa de redes de telecomunicaciones reales. Se presentan instancias de tamaño nunca antes resueltas en la literatura.
- Propuesta de un nuevo problema de optimización en redes *ad hoc* de dispositivos móviles (MANETs) en el que el proceso de difusión (*broadcasting*) se formula como problema de optimización multiobjetivo. Creación de un cuerpo de conocimiento sobre este problema.
- Desarrollo de nuevos modelos algorítmicos para la resolución de problemas de optimización multiobjetivo:
  - AbYSS (*Archive-based hYbrid Scatter Search*): algoritmo de optimización basado en búsqueda dispersa.
  - cMOGA (*cellular Multi-Objective Genetic Algorithm*) y MOCell (*Multi-Objective Cellular genetic algorithm*): dos algoritmos genéticos celulares para optimización multiobjetivo.
  - pPAES (*parallel PAES*): modelo paralelo para la estrategia evolutiva PAES (*Pareto Archived Evolution Strategy*), uno de los algoritmos básicos del campo de la optimización multiobjetivo.
  - ssNSGA-II (*steady state NSGA-II*): incorpora la selección por estado estacionario al algoritmo generacional NSGA-II (*Non-dominating Sorting Genetic Algorithm II*).
- Aplicación, por primera vez, del algoritmo de búsqueda dispersa a los problemas de la planificación de celdas (ACP) y asignación de frecuencias (AFP).
- Desarrollo de nuevos modelos de los algoritmos propuestos para su ejecución en sistemas de computación grid:
  - GrEA (*Grid-based Evolutionary Algorithm*): extensión del algoritmo genético con selección por estado estacionario.
  - assNSGA-II (*asynchronous ssNSGA-II*): extensión grid del algoritmo ssNSGA-II.
- Resolución, por primera vez, del problema AFP utilizando un sistema de computación grid.

Finalmente, se ha llevado a cabo una importante labor de diseminación del contenido de las investigaciones desarrolladas en este trabajo. Para tal fin, además de las publicaciones aparecidas en revistas y congresos, tanto nacionales como internacionales, se han desarrollado páginas web de dominio público con la descripción de los problemas y los resultados de las investigaciones.

1.4. Organización de la tesis

Este documento de tesis se estructura en cuatro partes. En la primera se presentan los conceptos de las metaheurísticas y la optimización en redes de telecomunicaciones. En la segunda parte se modelan los tres problemas considerados para su resolución posterior. La parte tercera incluye las propuestas metodológicas y sus extensiones avanzadas para multiobjetivo y grid. Se presentan,

además, los resultados de la aplicación de estas propuestas a los problemas de redes de telecomunicaciones formulados previamente. Finalmente, en la cuarta parte mostramos las principales conclusiones que se desprenden de este trabajo, así como las líneas de trabajo futuro inmediatas que surgen de nuestro estudio. A continuación describimos detalladamente el contenido de los capítulos.

- **Parte I: Fundamentos**

  El Capítulo 2 da una visión global sobre optimización en redes de telecomunicaciones e introduce los tres problemas resueltos en esta tesis: la planificación de celdas, la asignación de frecuencias y la optimización del proceso de difusión en redes ad hoc de dispositivos móviles.

  El Capítulo 3 proporciona una introducción genérica sobre el campo de las metaheurísticas. Posteriormente se dan detalles sobre los mecanismos avanzados para algoritmos metaheurísticos que se usan en esta tesis: optimización multiobjetivo, paralelismo y sistemas de computación grid.

- **Parte II: Modelado de problemas**

  Los Capítulos 4, 5 y 6 presentan, respectivamente, el problema de la planificación de celdas en redes de telefonía, tanto de segunda como de tercera generación, el problema de la asignación de frecuencias en redes de segunda generación y el problema de la difusión óptima en redes ad hoc de dispositivos móviles. En cada capítulo se discuten los trabajos relacionados con cada problema y se formulan los problemas dando modelos matemáticos muy precisos de cada uno de ellos.

- **Parte III: Diseño y evaluación de metaheurísticas para los problemas propuestos**

  El Capítulo 7 describe las propuestas metodológicas utilizadas para resolver los problemas seleccionados. Se detallan los algoritmos empleados y las modificaciones a éstos, así como los nuevos modelos desarrollados especialmente para resolver los problemas. Hablamos de extensiones avanzadas en dos sentidos: primero, para resolver problemas de optimización multiobjetivo y, segundo, para que se puedan ejecutar en sistemas de computación grid.

  El Capítulo 8 presenta los resultados obtenidos al resolver el problema de planificación de celdas en redes de telefonía. Se resuelven distintas instancias del problema con diferentes características y se discuten los resultados para llegar a una comprensión profunda del problema que pueda servir a los diseñadores de este tipo de redes.

  El Capítulo 9 muestra y discute los resultados obtenidos tras la aplicación de las técnicas metaheurísticas al problema de la asignación de frecuencias. Se analizan los distintos algoritmos sobre diferentes instancias reales del problema.

  El Capítulo 10 estudia los resultados obtenidos para el problema de la optimización de la estrategia de difusión en redes ad hoc de dispositivos móviles. Se han resuelto dos formulaciones del problema sobre tres instancias diferentes que modelan tres escenarios reales.

- **Parte V: Conclusiones y trabajo futuro**

  Terminamos esta tesis con unas conclusiones sobre todo lo expuesto en el resto del documento. Asimismo, se describen también las principales líneas de trabajo futuro que surgen del presente estudio. Esta parte está redactada tanto en español como en inglés para cumplir los requisitos exigidos por la Universidad de Málaga para optar a la mención de Doctorado Europeo.

- **Parte VI: Apéndices**

  El Apéndice A presenta las publicaciones del doctorando realizadas durante la elaboración de la tesis y que sustentan la calidad de la misma.

  El Apéndice B contiene un resumen en inglés de la tesis doctoral, necesario para optar a la mención de Doctorado Europeo.

Parte I

Fundamentos

Capítulo 2

Optimización y redes de telecomunicaciones

Las telecomunicaciones han tenido un enorme impacto en prácticamente todos los aspectos de la vida en el último siglo. Existen pocas dudas de que la transformación de la edad industrial a la era de la información ha sido promovida, fundamentalmente, por los avances alcanzados en los sistemas de telecomunicaciones. Los problemas de optimización que aparecen en esta industria son muchos y su resolución satisfactoria ha jugado un papel muy importante tanto en el desarrollo como en el uso generalizado de estos sistemas. En general, muchos de los problemas están relacionados con la red de comunicación subyacente que se encarga de transmitir la información, tanto en su diseño como en el propio funcionamiento de la misma [219]. No obstante, existen otros aspectos comerciales, como la decisión de la tecnología que se va a utilizar, los posibles usuarios finales o el segmento del mercado al que se pretende dar servicio, que también involucran procesos de optimización y que no están dentro del ámbito en el que nos movemos en esta tesis.

Las redes de telecomunicaciones de hoy en día son cada vez más complejas, en esencia porque deben satisfacer un cada vez más amplio rango de requisitos, entre los que se encuentran la coexistencia de servicios y aplicaciones heterogéneos en la misma red, el soporte para permitir movilidad, o la necesidad de interconexión con otras redes (posiblemente de distinto tipo como, por ejemplo, red de telefonía fija y red de telefonía móvil). Como consecuencia, diseñar una red es también cada vez más difícil. Si bien la forma usual de abordar este problema consiste en dividir la tarea del diseño en subtareas más pequeñas y restringir algunas de las posibilidades de diseño, las subtareas resultantes siguen siendo problemas muy complejos (por ejemplo, el diseño de la topología, configuración de los componentes, encaminamiento, localización de funcionalidades y asignación de recursos) que resultan ser, de forma natural, problemas de optimización. De esta forma, a medida que las redes se han vuelto progresivamente más complejas, la ayuda de las técnicas de optimización también ha ido creciendo en importancia.

Uno de los sistemas de telecomunicaciones que se han desarrollado con mayor rapidez debido, en gran medida, a las oportunidades de negocio para las empresas del sector son, sin lugar a dudas, los sistemas de telefonía móvil. Esto ha hecho que la investigación en el campo sea muy intensa y ha desencadenado considerables avances tecnológicos así como multitud de modelos matemáticos y algoritmos de optimización para respaldar decisiones de planificación y gestión. La mayor contribución de la optimización aquí consiste en mejorar la forma en la que se asignan los recursos escasos de los que se dispone (espectro de transmisión, antenas, etc.) y en incrementar la calidad de los servicios proporcionados (como, por ejemplo, ancho de banda o retraso de las transmisiones). Los problemas de optimización más importantes que aparecen en los sistemas de telefonía

móvil tienen que ver con el propio diseño del sistema. De esta forma, diseñar una red de este tipo conlleva múltiples tareas de planificación [178] entre las que se encuentran la planificación de la red de radio, la planificación de la red de transmisión y la planificación de la red del núcleo (*radio network planning*, *transmission network planning* y *core network planning*, respectivamente). La tarea más importante en todo el proceso de diseño es, quizás, la planificación de la red de radio, ya que es la parte de la misma encargada de la conexiones inalámbricas con los usuarios móviles y es la que vamos a considerar en este trabajo. En este escenario, la tecnología subyacente va a determinar en gran medida el problema particular que se quiera resolver, por lo que es un factor clave que hay que considerar. Así, en esta tesis nos vamos a centrar en la planificación de la red de radio de sistemas de telefonía móvil de segunda generación o GSM (*Global System for Mobile communication*) [186]. El sistema de comunicaciones GSM es un estándar mundial para teléfonos móviles digitales que permite transmitir tanto voz como datos. La tecnología GSM también suele denominarse 2G (por ser la segunda generación de este tipo de tecnologías inalámbricas) y es sin lugar a dudas el sistema de comunicaciones de mayor éxito en el mundo. De hecho, a mediados de 2006 había más de 1.800 millones de abonados a servicios GSM en 210 países[1], lo que se corresponde aproximadamente con el 77 % del mercado mundial de telefonía móvil. También es comúnmente aceptado que UMTS (*Universal Mobile Telecommunication System*) [210], la tercera generación de sistemas de telecomunicaciones móviles, coexistirá con las implementaciones más avanzadas del estándar GSM (GPRS [103] y EDGE [92]), al menos en sus primeras fases. Por tanto, se espera que GSM siga jugando un papel muy importante como tecnología dominante por varios años.

En GSM, la planificación de la red de radio es tan compleja que se divide en dos subproblemas: la planificación de celdas o ACP (*Automatic Cell Planning*), que hace referencia a la localización de antenas y la configuración de las características de propagación de señales de las mismas, y la asignación de frecuencias o AFP (*Automatic Frequency Problem*), en donde, tomando como partida las celdas obtenidas en el ACP, un número de frecuencias han de ser asignadas a los transmisores de forma que se minimicen las interferencias en la red. La tecnología UMTS hace que esta subdivisión no sea necesaria y ambos problemas han de abordarse simultáneamente. En nuestro caso, no obstante, debido a la formulación utilizada para la planificación de celdas en GSM (ver Capítulo 4), las conclusiones extraídas para esta tecnología de segunda generación son también válidas para UMTS. ACP y AFP son los dos primeros problemas de optimización cuya resolución se aborda en esta tesis.

La discusión anterior pone de manifiesto que la complejidad asociada al diseño de redes de telefonía móvil es muy elevada, ya que cada vez se van incorporando demandas de nuevos servicios, tecnologías de última generación, etc. Existe, no obstante, una tendencia dentro de las telecomunicaciones a reducir la necesidad de planificación y configuración mediante el desarrollo de sistemas de bajo coste, baja complejidad y mínima configuración, dando lugar a lo que se denominan redes ad hoc de dispositivos móviles (MANETs o *Mobile Ad hoc NETworks*). Esta redes son sistemas autónomos o auto-organizados compuestos por un conjunto de dispositivos móviles conectados por un canal inalámbrico compartido donde, además, no existe infraestructura alguna. Debido a los límites físicos de los enlaces inalámbricos y la movilidad de los dispositivos, la topología de la red es muy dinámica y hace que las comunicaciones puedan ser tremendamente ineficientes. A diferencia de las redes de telefonía móvil, donde el diseño de las mismas proporcionaba los problemas de optimización más importantes, el hecho de no necesitar infraestructura previa hace que los problemas en MANETs surjan en la propia operación de la red como por ejemplo en el encaminamiento de paquetes o en el ahorro de energía debido a que los dispositivos tienen autonomía limitada. El trabajo en este campo se enfoca hacia las operaciones de difusión (o *broadcasting*), que consisten en enviar un paquete desde un dispositivo al resto de dispositivos de la red. Debido a las topologías tan dinámicas que aparecen, estas operaciones son muy frecuentes, ya que son el mecanismo básico

[1] http://www.wirelessintelligence.com/

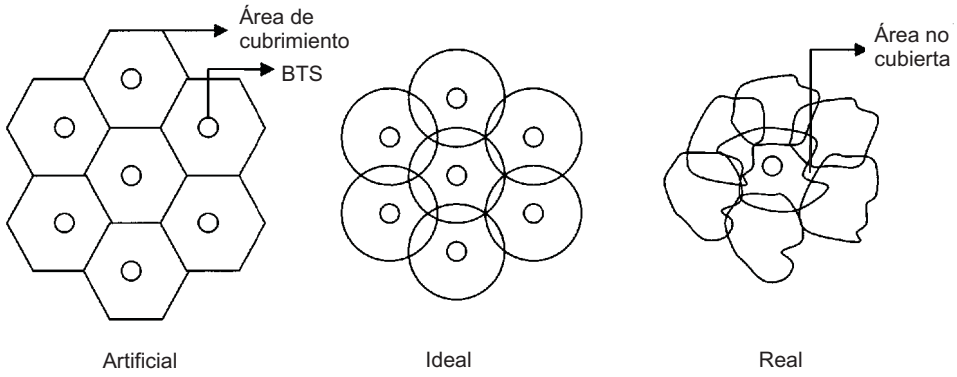


Figura 2.1: Forma de las celdas.

para resolver problemas de la propia red (como es el caso de algoritmos de encaminamiento). La elección de la estrategia de difusión apropiada va a tener, por tanto, un enorme impacto en el rendimiento global de la red. Éste es el tercer problema que se considera en esta tesis dentro del campo de las telecomunicaciones: el problema de la optimización de la estrategia de difusión en MANETs.

El resto del capítulo se organiza como sigue. En la siguiente sección se dan más detalles sobre los problemas de planificación de la red de radio en sistemas telefonía móvil, tanto ACP como de la AFP. El problema de la difusión óptima en MANETs se presenta en la Sección 2.2. En la Sección 2.3 se incluyen las conclusiones de este capítulo.

2.1. Planificación de la red de radio en sistemas de telefonía móvil

Los sistemas de telefonía móvil proporcionan servicios de telecomunicaciones mediante un conjunto de estaciones base (BTSs o *Base Transceiver Stations*) que son capaces de gestionar conexiones de radio con estaciones móviles (MSs o *Mobile Stations*) dentro de su área de servicio [249]. Este área, llamada celda, está compuesta por el conjunto de puntos donde la intensidad de la señal recibida de la BTS considerada es mayor que las que proceden de otras BTSs y además es superior a un determinado valor umbral que asegura el funcionamiento con una mínima calidad garantizada. El nivel de potencia recibida depende de la potencia de transmisión, de la atenuación debida a la propagación de la señal desde la fuente hasta el destino, así como de las características de la antena y sus parámetros de configuración tales como la máxima potencia de emisión, la altura, la orientación y el diagrama de radiación. Como resultado, la forma y tamaño de las celdas varía considerablemente dependiendo de la localización de las BTSs y de sus parámetros de configuración, así como de la propagación. La Figura 2.1 muestra la forma de las celdas cuando se modelan artificialmente, su forma ideal y la forma real que tienen en la red.

Cuando los usuarios se mueven dentro del área que atraviesa los límites de una celda, la continuidad del servicio está garantizada por procedimientos de *handover*, que son capaces de transferir dicho servicio de una BTS a otra cuando la calidad del enlace es suficiente. En algunos casos, se pueden usar varias conexiones simultáneas con dos o más BTSs para mejorar la eficiencia.

Para permitir muchas conexiones simultáneas entre BTSs y MSs, la banda de radio disponible para transmisiones se divide en canales de radio mediante técnicas de multiplexación de canales. En la mayoría de los sistemas de segunda generación como GSM, la banda de radio se divide primero en

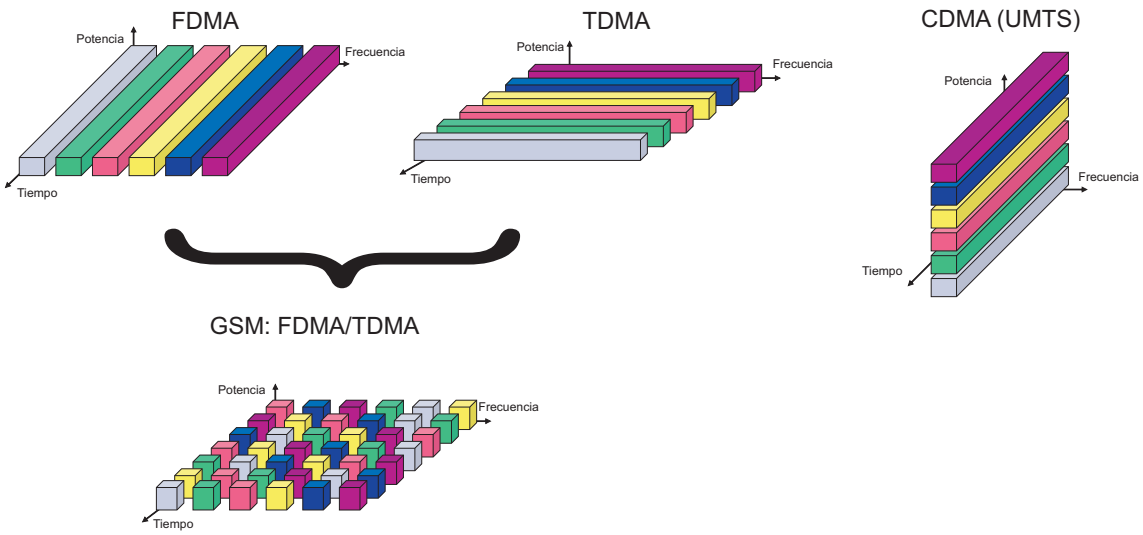


Figura 2.2: Técnicas de multiplexación: TDMA, FDMA, CDMA (UMTS) y combinación usada en GSM (TDMA/FDMA).

portadoras a diferentes frecuencias usando FDMA (*Frequency Division Multiple Access*) y, después, en cada portadora se crean canales utilizando TDMA (*Time Division Multiple Access*) [249]. La Figura 2.2 ilustra gráficamente estas técnicas de multiplexación. Con conexiones bidireccionales se necesita un par de canales sobre diferentes portadoras para las transmisiones de BTS a MS (*downlink*) y de MS a BTS (*uplink*), de acuerdo con el esquema FDD (*Frequency Division Duplexing*). Las BTSs puede utilizar multiples frecuencias mediante un conjunto de transmisores/receptores o TRXs (*transceivers*).

No obstante, el número de canales de radio que se obtienen de esta forma (varios cientos en sistemas de segunda generación) no son suficientes para dar servicio a grandes cantidades de usuarios. Para incrementar la capacidad del sistema estos canales de radio se han de reutilizar en diferentes celdas, lo que genera interferencias que pueden afectar a la calidad de las señales recibidas. Sin embargo, si el ratio entre la potencia de la señal recibida y las interferencias (suma de las potencias de las señales interferentes) es mayor que un determinado umbral, la señal se puede decodificar correctamente (*efecto captura*).

Para garantizar que esta condición se satisface durante todas las operaciones del sistema, la asignación de canales a BTSs se debe realizar cuidadosamente. Obviamente, cuanto más densa sea la reutilización de canales, mayor será el número de canales disponibles por celda y se podrá dar servicio a un mayor número de transmisiones. Por tanto, la asignación de canales determina la capacidad del sistema. Aunque la mayor fuente de interferencia aparece en las 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.

Planificar un sistema de telefonía móvil implica seleccionar las localizaciones donde se instalarán las BTSs, establecer sus parámetros de configuración (potencia de emisión, altura de la antena, ángulo de inclinación o *tilt*, el acimut o *azimuth*, etc.) y asignar frecuencias de forma que se cubra todo el área y se garantice una capacidad suficiente para cada celda. Debido a la complejidad del problema, en los sistemas de segunda generación se adopta una aproximación en dos fases. En primer lugar se planifica el cubrimiento para garantizar que se recibe un nivel suficiente de

señal en todo el área de servicio desde al menos una BTS. Entonces, las frecuencias disponibles se asignan a las BTSs considerando las interferencias y las restricciones de capacidad. Para los sistemas de tercera generación, esta aproximación en dos fases no es apropiada, ya que utiliza W-CDMA (*Wideband Code Division Multiple Access*) [115] y el ancho de banda está compartido por todas las transmisiones, por lo que la asignación de frecuencias no se necesita (Figura 2.2). La capacidad del sistema depende de los niveles de interferencias existentes en la red. Como estos valores dependen, a su vez, de la distribución de tráfico así como de la localización y configuración de BTSs, la cobertura (planificación de celdas) y la capacidad (asignación de frecuencias) deben planificarse conjuntamente.

2.1.1. Planificación de celdas

El problema general de la planificación de celdas (o planificación de cobertura) se puede describir como sigue. Dada un área donde se ha de garantizar el servicio, hay que determinar dónde se colocan las BTSs y seleccionar la configuración de las mismas de forma que cada punto (o cada usuario) del área de servicio reciba un nivel de señal suficientemente alto. Ya que el coste asociado a cada BTS suele depender de su localización y configuración, un objetivo típico consiste en minimizar el coste total de instalación de antenas a la vez que se garantiza el cubrimiento.

Como se viene discutiendo hasta ahora, la planificación de celdas implica resolver dos tareas de optimización. Hay que seleccionar, primero, un conjunto de localizaciones para las BTSs de entre un conjunto de sitios candidatos. Una vez seleccionadas estas localizaciones, las BTSs que ahí se coloquen se deben configurar de forma que proporcionen la cobertura y capacidad adecuadas, a la vez que cumplan con un conjunto de restricciones definidas para garantizar un mínimo de calidad en las comunicaciones. La Figura 2.3 muestra gráficamente un ejemplo de área geográfica a la que se debe dar servicio, el conjunto de sitios candidatos que existen en la red y los parámetros típicos de una BTS. Configurar de forma eficiente las BTSs (potencia, inclinación, acimut, etc.) es una tarea difícil no sólo por el número potencial de configuraciones, sino porque cambiar la configuración de una BTS puede afectar a otras BTSs. Así, si se reduce la potencia de una BTS para reducir la cantidad de tráfico al que da servicio, otra u otras BTSs deben dar servicio a ese tráfico por lo que, a su vez, se pueden sobrecargar. Esto implica, además, un elevado coste computacional ya que, haciendo incluso cambios locales pequeños, supone recalcular medidas por toda la red.

En la literatura se han definido diferentes modelos para resolver el problema real de la planificación de celdas que van desde los puramente abstractos (por ejemplo, modelos basados en teoría de grafos) hasta los más detallados (modelos que consideran áreas específicas con tráfico y topologías conocidas). Los modelos intermedios, bien tratan de añadir realismo a un modelo abstracto, bien reducen la complejidad de los modelos detallados para aliviar la carga computacional que supone. Los tres principales modelos que existen en la literatura se describen a continuación.

Modelo de nodos de demanda

El concepto de nodos de demanda se introdujo por primera vez en [94] y se ha utilizado en diferentes trabajos desde entonces, como por ejemplo en [18, 93, 239, 250]. El concepto básico es que el nodo de demanda representa el centro de un área donde el tráfico está siendo generado por los usuarios. La principal ventaja de este modelo es que, al combinar el tráfico de una área geográfica pequeña en un sólo punto, la carga computacional se reduce enormemente, si bien el realismo del problema también se reduce. Por ejemplo, en [250] sólo se necesitaban 288 nodos de demanda para un área de 81 km^2, mientras que en un modelo discreto con puntos de test uniformemente distribuidos cada 100 metros (ver Sección 2.1.1) serían 2025 nodos. En efecto, hay muchos menos nodos de demanda, ya que cada uno comprende un número de puntos de test. Sin

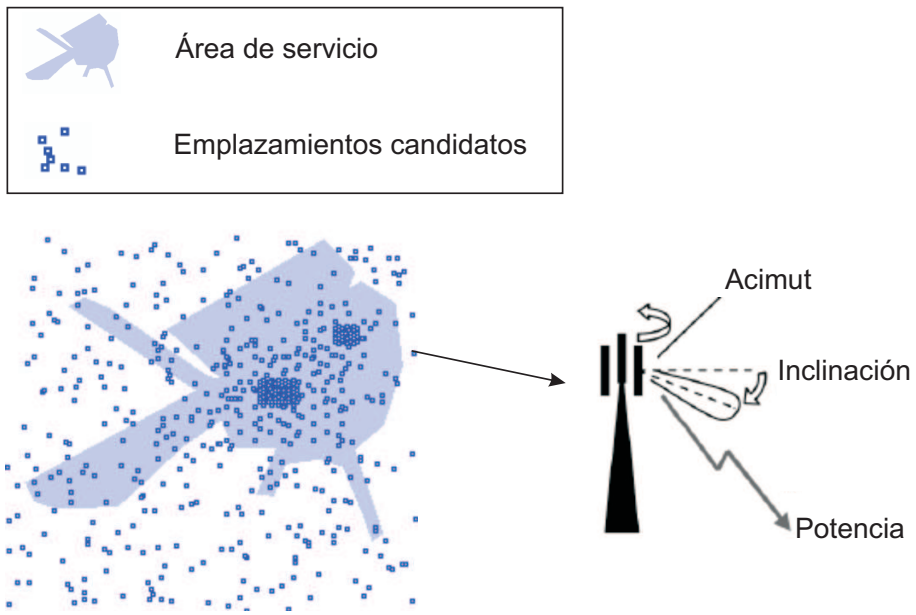


Figura 2.3: Ejemplo de área geográfica a la que hay que dar servicio, conjunto de sitios candidatos y parámetros de configuración típicos de cada BTS.

embargo, el proceso de fusionar varios puntos de test en un nodo de demanda tiene el mismo efecto que cualquier tipo de compresión: la resolución o claridad disminuyen.

Los investigadores que han utilizado este modelo de nodos de demanda hasta la fecha han permitido total libertad para seleccionar los sitios candidatos. Esto abre la posibilidad de distribuir uniformemente las BTSs por todo el área a cubrir, algo que generalmente no está permitido y, en general, no es posible de hacer por parte de los operadores.

Modelo de discos

La aplicación de gráficos de disco (circulares) en el diseño de sistemas celulares se usó por primera vez en [105] para resolver el problema de la asignación de frecuencias. Existen extensiones posteriores de este modelo en las que se consideran intersecciones entre discos y distribuciones de tráfico no uniformes [116, 117, 118]. El punto fuerte de la aproximación presentada en [116], por ejemplo, es que permite considerar diferentes objetivos de diseño de red, lo que significa que se puede resolver los problemas de la planificación de celdas y la asignación de frecuencias simultáneamente. La carga computacional que exige es también baja.

Sin embargo, el modelo tiene algunos inconvenientes claros. En primer lugar, se asume un modelo de propagación idealizado, por lo que todas las celdas tiene idéntica forma. Si bien el tamaño de estas celdas puede cambiar atendiendo a una distribución de tráfico no uniforme [117], la forma de las mismas es siempre la misma (circular). Otro problema que aparece es que las BTSs se pueden situar en cualquier localización, por lo que surgen los mismos problemas que en el modelo anterior.

Modelo de celdas y puntos de test

Aunque su mejor definición aparece quizás en [215, 216, 217], este modelo ha aparecido con un formato similar en trabajos previos de la literatura [109]. En este modelo, el área de trabajo

se discretiza en un conjunto de puntos de test repartidos por todo el área. Estos puntos de test se usan para permitir la medida de la potencia de la señal en el área donde el operador concreto desea dar servicio a la demanda de tráfico de una serie de clientes. Se definen tres subconjuntos de puntos de test repartidos en el área de trabajo: los puntos de recepción (*Reception Test Points* o RTPs), donde se prueba la recepción de la señal; los puntos de servicio (*Service Test Points* o STPs), donde la calidad de la señal debe superar un umbral mínimo para que sea utilizable por los clientes; y los puntos de tráfico (*Traffic Test Points* o TTPs), donde se asocia cierta cantidad de tráfico de los usuarios (en erlangs). El modelo garantiza que TTP $\subseteq$ STP $\subseteq$ RTP.

También define un conjunto de localizaciones candidatas para BTSs que no están distribuidas uniformemente en el terreno, de forma que se refleja con mucho mayor realismo los escenarios que presentan los operadores. La principal ventaja de este modelo es que permite medir todos los objetivos de la red (como cobertura y capacidad). Hay, no obstante, una desventaja clara aquí y es el coste computacional que se necesita, ya que se suele utilizar mucha resolución a la hora de abordar el problema (por ejemplo, puntos de test cada 200 metros) para incrementar el realismo.

Precisamente porque permite modelar de forma precisa los problemas que se encuentran los operadores en la realidad, este modelo ha sido utilizado ampliamente en la literatura [119, 208, 207, 206, 236, 237, 245, 261]. Por su realismo y la disponibilidad de datos de redes GSM reales, éste es el modelo que hemos usado para resolver el problema de la planificación de frecuencias en esta tesis doctoral.

2.1.2. Asignación de frecuencias

En los sistemas de segunda generación, después de planificar las celdas, las frecuencias (o portadoras) disponibles han de asignarse a las BTSs para proporcionarles una capacidad suficiente que cubra las demandas de tráfico. La explotación eficiente del espectro de radio se consigue con la reutilización de frecuencias en la red. Sin embargo, esta reutilización de frecuencias puede deteriorar la calidad de las señales debido a las interferencias que aparecen en la red. La Figura 2.4 muestra un ejemplo de esto. La BTS A no provoca interferencias en la red ya que no existe solapamiento de su área de cobertura. Sin embargo, si a las BTSs B y C se les asigna la misma frecuencia o frecuencias adyacentes, sí que aparecen dichas interferencias.

El problema de la asignación de frecuencias es, por tanto, el problema de asignar frecuencias a cada transmisor de una red inalámbrica de forma que alguna medida de la calidad de las señales recibidas es maximizada. Dependiendo del tamaño del espectro disponible, de los objetivos y de las restricciones tecnológicas existentes, este problema puede asumir muchas formas diferentes.

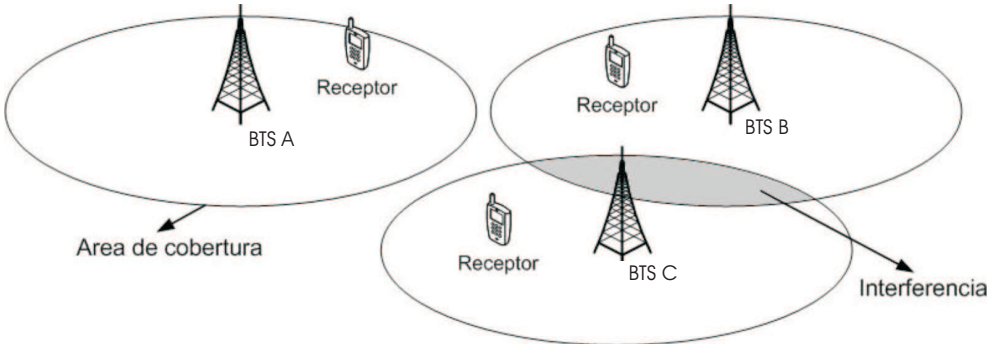


Figura 2.4: Red de ejemplo con 3 BTSs donde puede haber interferencias entre la BTS B y la BTS C, si utilizan frecuencias iguales o adyacentes.

Hay que decir que este problema es probablemente el que más atracción ha atraído en el campo de la Investigación Operativa, tanto por su relevancia como por su inmediata relación respecto a problemas clásicos de optimización combinatoria. Esta amplia producción bibliográfica ha sido analizada y organizada en varios libros y artículos de revisión [1, 80, 128, 148].

En los años 70, los gobiernos daban licencias de frecuencias en unidades; así, ya que se pagaba por cada frecuencia, los operadores trataron de minimizar el número total de frecuencias que se necesitaban para conseguir configuraciones sin interferencias. Se entendió rápidamente [176] que esto se correspondía con la resolución de una versión apropiada del problema de coloreado de grafos, o alguna generalización del mismo. En esta primera aproximación se asumía que frecuencias distintas no interferían entre ellas, es decir, sólo interferencias co-canal eran consideradas. Este problema se conoce como MO-FAP (*Minimum Order Frequency Assignment Problem*) [1]. Dado que, en general, éste no siempre es el caso, a principio de los 80 fueron apareciendo sucesivas generalizaciones de este modelo [105] que solventaban este inconveniente incorporando un nuevo parámetro para considerar interferencias entre frecuencias adyacentes. Además, los reguladores nacionales comenzaron a vender las frecuencias en bloques en lugar de por unidades, por lo que surgió una nueva versión del problema, conocida como MS-FAP (*Minimum Span Frequency Assignment Problem*) [1], dónde hay que minimizar el rango de frecuencias utilizadas (frecuencia máxima menos frecuencia mínima). Ambas versiones ha sido abordadas ampliamente en la literatura con técnicas que van desde simples generalizaciones de heurísticas clásicas de coloreado de grafos [30, 51] como DSATUR [31] hasta algoritmos avanzados de optimización como por ejemplo recocido simulado [51], algoritmos genéticos [241] o búsqueda tabú [108].

El incesante incremento en la demanda de tráfico de finales de los 80 y los 90 hizo que el espectro disponible fuese insuficiente en poco tiempo para obtener planes de frecuencia libres de interferencias. El objetivo de las planificación pasó, por tanto, de minimizar el número de frecuencias a maximizar la calidad del servicio, lo que se traduce a minimizar las interferencias de la red. A partir de aquí surgió el llamado MI-FAP (*Minimum Interference Frequency Assignment Problem* o problema de la asignación de frecuencias con mínimas interferencias). El modelo MI-FAP es, sin lugar a dudas, la versión del problema con la que se ha trabajado más en la literatura reciente, principalmente por su aplicabilidad directa a la resolución de instancias reales en redes GSM. Así, se han propuesto un número enorme de aproximaciones se han propuesto los últimos años [1]. El problema que abordamos en esta tesis se encuadra dentro de esta versión del problema.

2.2. Optimización del proceso de difusión en MANETs

Una red ad hoc de dispositivos móviles o MANET (*Mobile Ad hoc NETwork*) es una red creada sobre la marcha (*on the fly*) por dispositivos móviles autónomos que están equipados con tecnologías inalámbricas de corto alcance (ordenadores portátiles, teléfonos móviles, PDAs, etc.). Estos dispositivos funcionan independientemente unos de otros, aunque deben coordinarse y cooperar entre ellos para auto-organizar y auto-gestionar la red. Creadas inicialmente en investigaciones militares, las redes ad hoc están ganando importancia tanto en aplicaciones comerciales y académicas como en servicios de emergencia.

En general, las MANETs utilizan un único canal de radio y sus dispositivos pueden comunicarse directamente con aquellos otros que están dentro de su área de transmisión. Dado que no existe control centralizado o infraestructura fija, como estaciones base, para dar soporte a las comunicaciones, cuando un dispositivo quiere enviar un mensaje a otro que está fuera de su área de transmisión, hay que establecer comunicaciones en múltiples saltos (*multi-hop*), por lo que los propios dispositivos han de actuar como encaminadores de mensajes. El mayor desafío que presentan, no obstante, es la movilidad de los dispositivos, ya que conocer la topología de la red es prohibitivo y, por tanto, el coste de las comunicaciones es generalmente muy alto.

El *broadcasting* o difusión es una primitiva de comunicación que trata de enviar un mensaje desde un dispositivo fuente hasta todos los dispositivos de una red. En MANETs, el broadcasting es un mecanismo crítico en diferentes aplicaciones a distintos niveles, como la difusión de información o el mantenimiento de la información global de la red. Es también un mecanismo necesario para protocolos de encaminamiento en MANETs. Así, protocolos como DSR (*Dynamic Source Routing*), AODV (*Ad Hoc On Demand Distance Vector*), ZRP (*Zone Routing Protocol*) y LAR (*Location Aided Routing*) utilizan algún mecanismo de difusión o una derivación de éstos para establecer rutas. Los mecanismos de difusión se utilizan frecuentemente, además, para resolver otros problemas propios de la red, como la localización de dispositivos o el envío de señales de alarma. En entornos muy dinámicos con topologías que cambian rápidamente, el broadcasting es un método muy robusto para realizar otras primitivas de comunicación como multicast.

Los protocolos de difusión tradicionales para redes cableadas siguen principalmente una aproximación basada en árboles que no son adecuadas para MANETs, debido a las topologías dinámicas y al particionamiento que muchas veces aparece en este tipo de redes. Todo esto ha impulsado una amplia labor de investigación [232, 256] que ha resultado en un conjunto de propuestas de esquemas de difusión que tratan de optimizar algún aspecto concreto de la operación, como minimizar el número de retransmisiones, minimizar el consumo global de los nodos transmisores, minimizar el retraso total de la difusión, etc. En general, muchas de estas propuestas son estrategias genéricas cuyo objetivo es comportarse adecuadamente en la mayoría de las situaciones posibles. No obstante, los posibles escenarios de MANETs son muy diversos y existen otras aproximaciones que, en lugar de utilizar estrategias generales de difusión, desarrollan protocolos específicos que consideran las particularidades del escenario concreto para optimizar su comportamiento. Ejemplos de estos escenarios son las VANETs (*Vehicular Ad hoc NETworks*) [83], que son redes de topología muy dinámica con dispositivos en los que la energía no es un problema, o redes de sensores [76], caracterizadas por una movilidad nula o muy escasa de los dispositivos que, al estar alimentados por baterías de capacidad limitada, tienen que ahorrar toda la energía posible.

Siguiendo esta línea de investigación, el protocolo llamado Inundación con Retrasos y Vecindarios Acumulativos (o *Delayed Flooding with Cumulative Neighborhood* – DFCN) [112] es un ejemplo de estrategia diseñada específicamente para MANETs *Metropolitanas* [48]. Las MANETs metropolitanas se caracterizan por tener una densidad de nodos muy heterogénea y dinámica en las que las zonas de alta densidad no se mantienen activas durante todo el tiempo. Esto lleva a topologías compuestas por subconjuntos de redes ad hoc que pueden unirse y separarse dinámicamente durante la propia operación de difusión. Escenarios claros de este tipo de redes son aeropuertos, estaciones de tren o edificios de oficinas.

Supongamos que una cadena de centros comerciales pretende introducir en sus edificios un servicio con el que, la gente que disponga de teléfonos móviles, PDAs, etc., pueda recibir eventualmente pequeños anuncios procedentes de las propias tiendas del edificio en los que publicitan sus productos, ofertas especiales, etc. En este escenario tan concreto sería deseable encontrar la mejor estrategia posible, cualquiera que sea el protocolo de difusión empleado (por ejemplo, DFCN). Éste es el enfoque seguido en este estudio: optimizar el servicio de difusión de mensajes para una red concreta [11, 160]. En nuestro caso, la estrategia de broadcasting considerada para su optimización es DFCN y los escenarios objetivo donde se despliega son MANETs metropolitanas. Dado que disponer de este tipo de redes no es fácil, hemos tenido que realizar simulaciones para diseñar y evaluar los escenarios utilizados. Este problema, que se ha definido por primera vez durante el desarrollo de esta tesis, es, hasta donde conocemos, el primer intento en el que el ajuste de una estrategia de difusión para un escenario concreto en MANETs metropolitanas se formula como un problema de optimización.

2.3. Conclusiones

En este capítulo hemos revisado algunos de los problemas de optimización más importantes que aparecen en el campo de las redes de telecomunicaciones móviles. La discusión se ha centrado en dos sistemas de telecomunicaciones: las redes de telefonía celular y las redes ad hoc de dispositivos móviles (MANETs). El objetivo ha sido introducir los tres problemas abordados en esta tesis así como dar una breve revisión de los mismos.

Dentro de las redes de telefonía móvil, los problemas de optimización más importantes que surgen tienen que ver con el propio diseño de la red. En concreto, hemos estudiado dos problemas relacionados con el diseño de la red de radio de segunda generación o GSM. La red de radio es considerada como la más importante ya que es la más cercana a los usuarios finales. En este contexto aparecen el problema de la planificación de celdas, o cómo posicionar y configurar las antenas para garantizar una señal con una calidad mínima en toda la red, y de la asignación de frecuencias, que consiste en asignar el espectro de radio disponible a los transmisores/receptores de la red.

También se ha discutido el problema del diseño de una estrategia de difusión óptima para MANETs. Este tipo de redes están consideradas como el futuro de las telecomunicaciones. Debido a sus características (movilidad, dinamismo, etc.), las operaciones de difusión son muy frecuentes en este tipo de redes y, por tanto, tener una estrategia ajustada a la red concreta va a influir muy notablemente en su rendimiento. Aquí es donde aparece el problema de optimización: obtener la mejor estrategia posible para un escenario dado. En este caso, la literatura es escasa y prácticamente todo el cuerpo de conocimiento sobre el problema se ha desarrollado a lo largo de esta tesis.

Capítulo 3

Metaheurísticas

Este capítulo está dedicado a establecer los fundamentos necesarios sobre los algoritmos de optimización utilizados para abordar los tres problemas del campo de las telecomunicaciones presentados en el capítulo anterior. Partiremos de un planteamiento clásico de optimización para definir el concepto de metaheurística y establecer su clasificación. A continuación se introducen conceptos de optimización multiobjetivo, puesto que tratamos con problemas de ingeniería donde existen varias funciones que se han de optimizar a la vez. Estos problemas no sólo proceden del mundo real, sino que además estamos considerando instancias reales (de gran tamaño) que suponen tareas con enormes necesidades de cómputo. Hemos utilizado técnicas paralelas para afrontar este inconveniente. Esto nos lleva al siguiente bloque de este capítulo, los modelos paralelos para metaheurísticas como medio para reducir los tiempos de ejecución. Por último, terminamos con el procedimiento estadístico seguido para evaluar metaheurísticas, donde se presentan las principales medidas de rendimiento así como los indicadores de calidad utilizados tanto en problemas de optimización monoobjetivo como multiobjetivo.

3.1. Definición de metaheurística

La optimización en el sentido de encontrar la mejor solución, o al menos una solución lo suficientemente buena, para un problema es un campo de vital importancia en el mundo real y, en particular, en ingeniería. Constantemente estamos resolviendo pequeños problemas de optimización, como el camino más corto para ir de un lugar a otro, la organización de una agenda, etc. En general, estos problemas son lo suficientemente pequeños y podemos resolverlos sin ayuda adicional, pero conforme se hacen más grandes y complejos, el uso de los ordenadores para su resolución es inevitable.

Comenzaremos este capítulo dando una definición formal del concepto de optimización. Asumiendo, sin perdida de generalidad, el caso de la minimización, podemos definir un *problema de optimización* como sigue:

Definición 1 (Problema de optimización)**.** *Un problema de optimización se formaliza como un par* (S, f)*, donde* $S \neq \emptyset$ *representa el espacio de soluciones (o de búsqueda) del problema, mientras que* f *es una función denominada* función objetivo *o* función de fitness*, que se define como:*

$$f : S \rightarrow \mathbb{R} \ . \tag{3.1}$$

Así, resolver un problema de optimización consiste en encontrar una solución, $i^* \in S$*, que satisfaga la siguiente desigualdad:*

$$f(i^*) \leq f(i), \quad \forall \ i \in S \ . \tag{3.2}$$

Asumir el caso de maximización o minimización no restringe la generalidad de los resultados, puesto que se puede establecer una igualdad entre tipos de problemas de maximización y minimización de la siguiente forma [19, 101]:

$$\text{máx}\{f(i)|i \in S\} \equiv \text{mín}\{-f(i)|i \in S\} \ . \tag{3.3}$$

En función del dominio al que pertenezca S, podemos definir problemas de *optimización binaria* ($S \subseteq \mathbb{B}^*$), *entera* ($S \subseteq \mathbb{N}^*$), *continua*($S \subseteq \mathbb{R}^*$), o *heterogénea* ($S \subseteq (\mathbb{B} \cup \mathbb{N} \cup \mathbb{R})^*$).

Debido a la gran importancia de los problemas de optimización, a lo largo de la historia de la Informática se han desarrollado múltiples métodos para tratar de resolverlos. Una clasificación muy simple de estos métodos se muestra en la Figura 3.1. Inicialmente, las técnicas las podemos clasificar en exactas (o enumerativas, exhaustivas, etc.) y aproximadas. Las técnicas exactas garantizan encontrar la solución óptima para cualquier instancia de cualquier problema en un tiempo acotado. El inconveniente de estos métodos es que el tiempo y/o memoria que se necesitan, aunque acotados, crecen exponencialmente con el tamaño del problema, ya que la mayoría de éstos son NP-duros. Esto supone en muchos casos que el uso de estas técnicas sea inviable, ya que se requiere mucho tiempo (posiblemente miles de años) y/o una cantidad desorbitada de memoria para la resolución del problema. Por lo tanto, los algoritmos aproximados para resolver estos problemas están recibiendo una atención cada vez mayor por parte de la comunidad internacional desde hace unas décadas. Estos métodos sacrifican la garantía de encontrar el óptimo a cambio de encontrar una solución satisfactoria en un tiempo razonable.

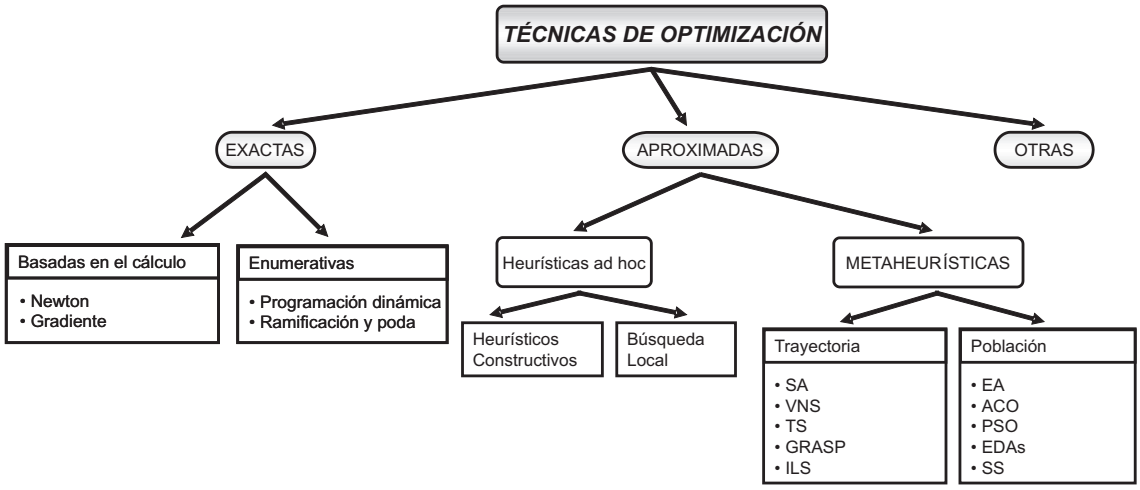


Figura 3.1: Clasificación de las técnicas de optimización.

Dentro de los algoritmos aproximados se pueden encontrar dos tipos: los heurísticos *ad hoc* y las metaheurísticas (en las que nos centramos en este capítulo). Los heurísticos *ad hoc*, a su vez, pueden dividirse en *heurísticos constructivos* y *métodos de búsqueda local*.

Los heurísticos constructivos suelen ser los métodos más rápidos. Construyen una solución desde cero mediante la incorporación de componentes hasta obtener una solución completa, que es el resultado del algoritmo. Aunque en muchos casos encontrar un heurístico constructivo es relativamente sencillo, las soluciones ofrecidas suelen ser de muy baja calidad. Encontrar métodos

de esta clase que produzca buenas soluciones es muy difícil, ya que dependen mucho del problema, y para su planteamiento se debe tener un conocimiento muy extenso del mismo. Por ejemplo, en problemas con muchas restricciones puede que la mayoría de las soluciones parciales sólo conduzcan a soluciones no factibles.

Los métodos de búsqueda local o seguimiento del gradiente parten de una solución ya completa y, usando el concepto de *vecindario*, recorren parte del espacio de búsqueda hasta encontrar un *óptimo local*. El vecindario de una solución s, que denotamos con $N(s)$, es el conjunto de soluciones que se pueden construir a partir de s aplicando un operador específico de modificación (generalmente denominado *movimiento*). Un óptimo local es una solución mejor o igual que cualquier otra solución de su vecindario. Estos métodos, partiendo de una solución inicial, examinan su vecindario y se quedan con el mejor vecino, continuando el proceso hasta que encuentran un óptimo local. En muchos casos, la exploración completa del vecindario es inabordable y se siguen diversas estrategias, dando lugar a diferentes variaciones del esquema genérico. Según el operador de movimiento elegido, el vecindario cambia y el modo de explorar el espacio de búsqueda también, pudiendo simplificarse o complicarse el proceso de búsqueda.

Finalmente, en los años setenta surgió una nueva clase de algoritmos aproximados, cuya idea básica era combinar diferentes métodos heurísticos a un nivel más alto para conseguir una exploración del espacio de búsqueda de forma eficiente y efectiva. Estas técnicas se han denominado *metaheurísticas*. Este término fue introducido por primera vez por Glover [96]. Antes de que el término fuese aceptado completamente por la comunidad científica, estas técnicas eran denominadas *heurísticas modernas* [214]. Esta clase de algoritmos incluye técnicas como colonias de hormigas, algoritmos evolutivos, búsqueda local iterada, enfriamiento simulado y búsqueda tabú. Se pueden encontrar revisiones de metaheurísticas en [29, 100]. De las diferentes descripciones de metaheurísticas que se encuentran en la literatura se pueden destacar ciertas propiedades fundamentales que caracterizan a este tipo de métodos:

- Las metaheurísticas son estrategias o plantillas generales que guían el proceso de búsqueda.
- El objetivo es una exploración eficiente del espacio de búsqueda para encontrar soluciones (casi) óptimas.
- Las metaheurísticas son algoritmos no exactos y generalmente son no deterministas.
- Pueden incorporar mecanismos para evitar regiones no prometedoras del espacio de búsqueda.
- El esquema básico de cualquier metaheurística tiene una estructura predefinida.
- Las metaheurísticas pueden hacer uso de conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados por la estrategia de más alto nivel.

Resumiendo estos puntos, se puede acordar que una metaheurística es una estrategia de alto nivel que usa diferentes métodos para explorar el espacio de búsqueda. En otras palabras, una metaheurística es una plantilla general no determinista que debe ser rellenada con datos específicos del problema (representación de las soluciones, operadores para manipularlas, etc.) y que permiten abordar problemas con espacios de búsqueda de gran tamaño. En este tipo de técnicas es especialmente importante el correcto equilibrio (generalmente dinámico) que haya entre *diversificación* e *intensificación*. El término diversificación se refiere a la evaluación de soluciones en regiones distantes del espacio de búsqueda (de acuerdo a una distancia previamente definida entre soluciones); también se conoce como *exploración* del espacio de búsqueda. El término intensificación, por otro lado, se refiere a la evaluación de soluciones en regiones acotadas y pequeñas con respecto al espacio de búsqueda centradas en el vecindario de soluciones concretas (*explotación* del espacio de búsqueda). El equilibrio entre estos dos aspectos contrapuestos es de gran importancia, ya que por

un lado deben identificarse rápidamente las regiones prometedoras del espacio de búsqueda global y por otro lado no se debe malgastar tiempo en las regiones que ya han sido exploradas o que no contienen soluciones de alta calidad.

Dentro de las metaheurísticas podemos distinguir dos tipos de estrategias de búsqueda. Por un lado, tenemos las extensiones "inteligentes" de los métodos de búsqueda local (metaheurísticas basadas en trayectoria en la Figura 3.1). La meta de estas estrategias es evitar de alguna forma los mínimos locales y moverse a otras regiones prometedoras del espacio de búsqueda. Este tipo de estrategia es el seguido por la búsqueda tabú, la búsqueda local iterada, la búsqueda con vecindario variable y el enfriamiento simulado. Estas metaheurísticas trabajan sobre una o varias estructuras de vecindario impuestas por el espacio de búsqueda. Otro tipo de estrategia es el seguido por las colonias de hormigas o los algoritmos evolutivos. Éstos incorporan un componente de aprendizaje en el sentido de que, de forma implícita o explícita, intentan aprender la correlación entre las variables del problema para identificar las regiones del espacio de búsqueda con soluciones de alta calidad (en la Figura 3.1, metaheurísticas basadas en población). Estos métodos realizan, en este sentido, un muestreo sesgado del espacio de búsqueda.

Más formalmente, una metaheurística se define como una tupla de elementos que, dependiendo de cómo se definan, dan lugar a una técnica concreta u otra. Esta definición formal ha sido desarrollada en [165] y posteriormente extendida en [43].

Definición 2 (Metaheurística)**.** *Una metaheurística $\mathcal{M}$ es una tupla formada por los siguientes ocho componentes:*

$$\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle \ , \tag{3.4}$$

donde:

- *$\mathcal{T}$ es el conjunto de elementos que manipula la metaheurística. Este conjunto contiene al espacio de búsqueda y en la mayoría de los casos coincide con él.*
- *$\Xi = \{(\xi_1, D_1), (\xi_2, D_2), \ldots, (\xi_v, D_v)\}$ es un conjunto de v pares. Cada par está formado por una variable de estado de la metaheurística y el dominio de dicha variable.*
- *μ es el número de soluciones con las que trabaja $\mathcal{M}$ en un paso.*
- *λ es el número de nuevas soluciones generadas en cada iteración de $\mathcal{M}$.*
- *$\Phi : \mathcal{T}^\mu \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\lambda \rightarrow [0,1]$ representa el operador que genera nuevas soluciones a partir de las existentes. Esta función debe cumplir para todo $x \in \mathcal{T}^\mu$ y para todo $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\lambda} \Phi(x,t,y) = 1 \ . \tag{3.5}$$

- *$\sigma : \mathcal{T}^\mu \times \mathcal{T}^\lambda \times \prod_{i=1}^{v} D_i \times \mathcal{T}^\mu \rightarrow [0,1]$ es una función que permite seleccionar las soluciones que serán manipuladas en la siguiente iteración de $\mathcal{M}$. Esta función debe cumplir para todo $x \in \mathcal{T}^\mu$, $z \in \mathcal{T}^\lambda$ y $t \in \prod_{i=1}^{v} D_i$,*

$$\sum_{y \in \mathcal{T}^\mu} \sigma(x,z,t,y) = 1 \ , \tag{3.6}$$

$$\begin{aligned} &\forall y \in \mathcal{T}^\mu, \sigma(x,z,t,y) = 0 \vee \\ &\quad \vee \sigma(x,z,t,y) > 0 \wedge \\ &(\forall i \in \{1,\ldots,\mu\} \bullet (\exists j \in \{1,\ldots,\mu\}, y_i = x_j) \vee (\exists j \in \{1,\ldots,\lambda\}, y_i = z_j)) \ . \end{aligned} \tag{3.7}$$

- $\mathcal{U} : \mathcal{T}^{\mu} \times \mathcal{T}^{\lambda} \times \prod_{i=1}^{v} D_i \times \prod_{i=1}^{v} D_i \rightarrow [0,1]$ *representa el procedimiento de actualización de las variables de estado de la metaheurística. Esta función debe cumplir para todo* $x \in \mathcal{T}^{\mu}$, $z \in \mathcal{T}^{\lambda}$ *y* $t \in \prod_{i=1}^{v} D_i$,

$$\sum_{u \in \prod_{i=1}^{v} D_i} \mathcal{U}(x,z,t,u) = 1 \ . \tag{3.8}$$

- $\tau : \mathcal{T}^{\mu} \times \prod_{i=1}^{v} D_i \rightarrow \{falso, cierto\}$ *es una función que decide la terminación del algoritmo.*

La definición anterior recoge el comportamiento estocástico típico de las técnicas metaheurísticas. En concreto, las funciones Φ, σ y $\mathcal{U}$ deben interpretarse como probabilidades condicionadas. Por ejemplo, el valor de $\Phi(x,t,y)$ se interpreta como la probabilidad de que se genere el vector de hijos $y \in \mathcal{T}^{\lambda}$ dado que actualmente el conjunto de individuos con el que la metaheurística trabaja es $x \in \mathcal{T}^{\mu}$ y su estado interno viene definido por las variables de estado $t \in \prod_{i=1}^{v} D_i$. Puede observarse que las restricciones que se le imponen a las funciones Φ, σ y $\mathcal{U}$ permite considerarlas como funciones que devuelven estas probabilidades condicionadas.

Definición 3 (Estado de una metaheurística)**.** *Sea* $\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle$ *una metaheurística y* $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{\mu}\}$ *el conjunto de variables que almacenarán las soluciones con las que trabaja la metaheurística. Utilizaremos la notación* $first(\Xi)$ *para referirnos al conjunto de las variables de estado de la metaheurística,* $\{\xi_1, \xi_2, \ldots, \xi_v\}$*. Un* estado *$s$ de la metaheurística es un par de funciones* $s = (s_1, s_2)$ *con*

$$s_1 : \Theta \rightarrow \mathcal{T}, \tag{3.9}$$

$$s_2 : first(\Xi) \rightarrow \bigcup_{i=1}^{v} D_i \ , \tag{3.10}$$

donde s_2 *cumple*

$$s_2(\xi_i) \in D_i \ \ \forall \xi_i \in first(\Xi) \ . \tag{3.11}$$

Denotaremos con $\mathcal{S}_{\mathcal{M}}$ *el conjunto de todos los estados de una metaheurística* $\mathcal{M}$*.*

Por último, una vez definido el estado de la metaheurística, podemos definir su dinámica.

Definición 4 (Dinámica de una metaheurística)**.** *Sea* $\mathcal{M} = \langle \mathcal{T}, \Xi, \mu, \lambda, \Phi, \sigma, \mathcal{U}, \tau \rangle$ *una metaheurística y* $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{\mu}\}$ *el conjunto de variables que almacenarán las soluciones con las que trabaja la metaheurística. Denotaremos con* $\overline{\Theta}$ *a la tupla* $(\theta_1, \theta_2, \ldots, \theta_{\mu})$ *y con* $\overline{\Xi}$ *a la tupla* $(\xi_1, \xi_2, \ldots, \xi_v)$*. Extenderemos la definición de estado para que pueda aplicarse a tuplas de elementos, esto es, definimos* $\overline{s} = (\overline{s}_1, \overline{s}_2)$ *donde*

$$\overline{s}_1 : \Theta^n \rightarrow \mathcal{T}^n \ , \tag{3.12}$$

$$\overline{s}_2 : first(\Xi)^n \rightarrow \left(\bigcup_{i=1}^{v} D_i \right)^n \ , \tag{3.13}$$

y además

$$\overline{s}_1(\theta_{i_1}, \theta_{i_2}, \ldots, \theta_{i_n}) = (s_1(\theta_{i_1}), s_1(\theta_{i_2}), \ldots, s_1(\theta_{i_n})) \ , \tag{3.14}$$

$$\overline{s}_2(\xi_{j_1}, \xi_{j_2}, \ldots, \xi_{j_n}) = (s_2(\xi_{j_1}), s_2(\xi_{j_2}), \ldots, s_2(\xi_{j_n})) \ , \tag{3.15}$$

para $n \geq 2$. Diremos que r es un estado sucesor de s si existe $t \in \mathcal{T}^{\lambda}$ tal que $\Phi(\overline{s}_1(\overline{\Theta}), \overline{s}_2(\overline{\Xi}), t) > 0$ y además

$$\sigma(\overline{s}_1(\overline{\Theta}), t, \overline{s}_2(\overline{\Xi}), \overline{r}_1(\overline{\Theta})) > 0 \;\; y \tag{3.16}$$

$$\mathcal{U}(\overline{s}_1(\overline{\Theta}), t, \overline{s}_2(\overline{\Xi}), \overline{r}_2(\overline{\Xi})) > 0 \;\; . \tag{3.17}$$

Denotaremos con $\mathcal{F}_{\mathcal{M}}$ la relación binaria "ser sucesor de" definida en el conjunto de estados de una metaheurística $\mathcal{M}$. Es decir, $\mathcal{F}_{\mathcal{M}} \subseteq \mathcal{S}_{\mathcal{M}} \times \mathcal{S}_{\mathcal{M}}$, y $\mathcal{F}_{\mathcal{M}}(s, r)$ si r es un estado sucesor de s.

Definición 5 (Ejecución de una metaheurística)**.** *Una ejecución de una metaheurística $\mathcal{M}$ es una secuencia finita o infinita de estados, $s_0, s_1, \ldots$ en la que $\mathcal{F}_{\mathcal{M}}(s_i, s_{i+1})$ para todo $i \geq 0$ y además:*

- *si la secuencia es infinita se cumple $\tau(s_i(\overline{\Theta}), s_i(\overline{\Xi})) = falso$ para todo $i \geq 0$ y*
- *si la secuencia es finita se cumple $\tau(s_k(\overline{\Theta}), s_k(\overline{\Xi})) = cierto$ para el último estado s_k y, además, $\tau(s_i(\overline{\Theta}), s_i(\overline{\Xi})) = falso$ para todo $i \geq 0$ tal que $i < k$.*

En las próximas secciones tendremos la oportunidad de comprobar cómo esta formulación general se puede adaptar a las técnicas concretas (obviando aquellos parámetros no fijados por la metaheurística o que dependen de otros aspectos como el problema o la implementación concreta).

3.2. Clasificación de las metaheurísticas

Hay diferentes formas de clasificar y describir las técnicas metaheurísticas [29]. Dependiendo de las características que se seleccionen se pueden obtener diferentes taxonomías: basadas en la naturaleza y no basadas en la naturaleza, con memoria o sin ella, con una o varias estructuras de vecindario, etc. Una de las clasificaciones más populares las divide en metaheurísticas *basadas en trayectoria* y *basadas en población*. Las primeras manipulan en cada paso un único elemento del espacio de búsqueda, mientras que las segundas trabajan sobre un conjunto de ellos (población). Esta taxonomía se muestra de forma gráfica en la Figura 3.2, que además incluye las técnicas más representativas. Estas metaheurísticas se describen en las dos secciones siguientes.

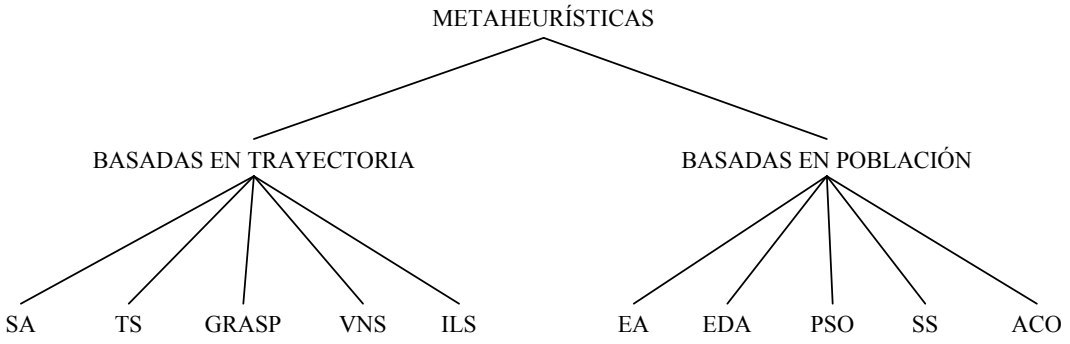


Figura 3.2: Clasificación de las metaheurísticas.

3.2.1. Metaheurísticas basadas en trayectoria

En esta sección repasaremos brevemente algunas metaheurísticas basadas en trayectoria. La principal característica de estos métodos es que parten de una solución y, mediante la exploración

del vecindario, van actualizando la solución actual, formando una trayectoria. Según la notación de la Definición 2, esto se formaliza con $\mu = 1$. La mayoría de estos algoritmos surgen como extensiones de los métodos simples de búsqueda local a los que se les añade algún mecanismo para escapar de los mínimos locales. Esto implica la necesidad de una condición de parada más elaborada que la de encontrar un mínimo local. Normalmente se termina la búsqueda cuando se alcanza un número máximo predefinido de iteraciones, se encuentra una solución con una calidad aceptable, o se detecta un estancamiento del proceso.

Enfriamiento simulado (SA)

El *enfriamiento simulado* o *Simulated Annealing* (SA) es una de las técnicas más antiguas entre las metaheurísticas y posiblemente es el primer algoritmo con una estrategia explícita para escapar de los mínimos locales. Los orígenes del algoritmo se encuentran en un mecanismo estadístico, denominado *metropolis* [175]. La idea del SA es simular el proceso de enfriamiento del metal y del cristal. El SA fue inicialmente presentado en [135]. Para evitar quedar atrapado en un mínimo local, el algoritmo permite elegir con cierta probabilidad una solución cuyo valor de *fitness* sea peor que el de la solución actual. En cada iteración se elige, a partir de la solución actual s, una solución s' del vecindario $N(s)$. Si s' es mejor que s (es decir, tiene un mejor valor en la función de *fitness*), se sustituye s por s' como solución actual. Si la solución s' es peor, entonces es aceptada con una determinada probabilidad que depende de la temperatura actual T y de la diferencia de *fitness* entre ambas soluciones, $f(s') - f(s)$ (caso de minimización).

Búsqueda tabú (TS)

La *búsqueda tabú* o *Tabu Search* (TS) es una de las metaheurísticas que se han aplicado con más éxito a la hora de resolver problemas de optimización combinatoria. Los fundamentos de este método fueron introducidos en [96], y están basados en las ideas formuladas en [95]. Un buen resumen de esta técnica y sus componentes se puede encontrar en [95].

La idea básica de la búsqueda tabú es el uso explícito de un historial de la búsqueda (una memoria a corto plazo), tanto para escapar de los mínimos locales como para implementar su estrategia de exploración y evitar buscar varias veces en la misma región. Esta memoria a corto plazo se implementa con una lista tabú, donde se mantienen las soluciones visitadas más recientemente para excluirlas de los próximos movimientos. En cada iteración se elige la mejor solución entre las permitidas y la solución es añadida a la lista tabú.

Desde el punto de vista de la implementación, mantener una lista de soluciones completas no suele ser práctico debido a su ineficiencia. Por lo tanto, en general, se suelen almacenar los movimientos que ha llevado al algoritmo a generar esa solución o los componentes principales que definen la solución. En cualquier caso, los elementos de esta lista permiten filtrar el vecindario, generando un conjunto reducido de soluciones elegibles denominado $N_a(s)$. El almacenamiento de los movimientos en vez de las soluciones completas es bastante más eficiente, pero introduce una perdida de información. Para evitar este problema, se define un criterio de aspiración que permite incluir una solución en $N_a(s)$ incluso si está prohibida debido a la lista tabú. El criterio de aspiración más ampliamente usado es permitir soluciones cuyo *fitness* sea mejor que el de la mejor solución encontrada hasta el momento.

GRASP

El *procedimiento de búsqueda miope aleatorizado y adaptativo* o *Greedy Randomized Adaptive Search Procedure* (GRASP) [84] es una metaheurística simple que combina heurísticos constructivos con búsqueda local. GRASP es un procedimiento iterativo, compuesto de dos fases: primero la

construcción de una solución y después un proceso de mejora. La solución mejorada es el resultado del proceso de búsqueda. El mecanismo de construcción de soluciones es un heurístico constructivo aleatorio. Va añadiendo paso a paso diferentes componentes c a la solución parcial s^p, que inicialmente está vacía. Los componentes que se añaden en cada paso son elegidos aleatoriamente de una lista restringida de candidatos (RCL). Esta lista es un subconjunto de $N(s^p)$, el conjunto de componentes permitidos para la solución parcial s^p. Para generar esta lista, los componentes de la solución en $N(s^p)$ se ordenan de acuerdo a alguna función dependiente del problema (η).

La lista RCL está compuesta por los α mejores componentes de ese conjunto. En el caso extremo de $\alpha = 1$, siempre se añade el mejor componente encontrado de manera determinista, con lo que el método de construcción es equivalente a un algoritmo voraz. En el otro extremo, con $\alpha = |N(s^p)|$ el componente a añadir se elige de forma totalmente aleatoria de entre todos los disponibles. Por lo tanto, α es un parámetro clave que influye en cómo se va a muestrear el espacio de búsqueda. La segunda fase del algoritmo consiste en aplicar un método de búsqueda local para mejorar la solución generada. Este mecanismo de mejora puede ser una técnica de mejora simple o algoritmos más complejos como SA o TS.

Búsqueda con vecindario variable (VNS)

La *búsqueda con vecindario variable* o *Variable Neighborhood Search* (VNS) es una metaheurística propuesta en [180] que aplica explícitamente una estrategia para cambiar entre diferentes vecindarios durante la búsqueda. Este algoritmo es muy general y con muchos grados de libertad a la hora de diseñar variaciones e instanciaciones particulares.

El primer paso a realizar es definir un conjunto de vecindarios. Esta elección puede hacerse de muchas formas: desde ser elegidos aleatoriamente hasta utilizar complejas ecuaciones deducidas del problema. Cada iteración consiste en tres fases: la elección del candidato, una fase de mejora y, finalmente, el movimiento. En la primera fase, se elige aleatoriamente un vecino s' de s usando el k-ésimo vecindario. Esta solución s' es utilizada como punto de partida de la búsqueda local de la segunda fase. Cuando termina el proceso de mejora, se compara la nueva solución s'' con la original s. Si es mejor, s'' se convierte en la solución actual y se inicializa el contador de vecindarios ($k \leftarrow 1$); si no es mejor, se repite el proceso pero utilizando el siguiente vecindario ($k \leftarrow k + 1$). La búsqueda local es el paso de intensificación del método y el cambio de vecindario puede considerarse como el paso de diversificación.

Búsqueda local iterada (ILS)

La *búsqueda local iterada* o *Iterated Local Search* (ILS) [156, 234] es una metaheurística basada en un concepto simple pero muy efectivo. En cada iteración, la solución actual es perturbada y, a esta nueva solución, se le aplica un método de búsqueda local para mejorarla. El mínimo local obtenido por el método de mejora puede ser aceptado como nueva solución actual si pasa un test de aceptación. La importancia del proceso de perturbación es obvia: si es demasiado pequeña puede que el algoritmo no sea capaz de escapar del mínimo local; por otro lado, si es demasiado grande, la perturbación puede hacer que el algoritmo sea como un método de búsqueda local con un reinicio aleatorio. Por lo tanto, el método de perturbación debe generar una nueva solución que sirva como inicio a la búsqueda local, pero que no debe estar muy lejos del actual para que no sea una solución aleatoria. El criterio de aceptación actúa como contra-balance, ya que filtra la aceptación de nuevas soluciones dependiendo de la historia de búsqueda y de las características del nuevo mínimo local.

3.2.2. Metaheurísticas basadas en población

Los métodos basados en población se caracterizan por trabajar con un conjunto de soluciones, usualmente denominado población, en cada iteración (es decir, generalmente $\mu > 1$ y/o $\lambda > 1$), a diferencia de los métodos basados en trayectoria, que únicamente manipulan una solución del espacio de búsqueda por iteración.

Algoritmos evolutivos (EA)

Los *algoritmos evolutivos* o *Evolutionary Algorithms* (EAs) están inspirados en la teoría de la evolución natural. Esta familia de técnicas sigue un proceso iterativo y estocástico que opera sobre una población de soluciones, denominadas en este contexto *individuos*. Inicialmente, la población es generada típicamente de forma aleatoria (quizás con ayuda de un heurístico de construcción).

El esquema general de un algoritmo evolutivo comprende tres fases principales: selección, reproducción y reemplazo. El proceso completo es repetido hasta que se cumpla un cierto criterio de terminación (normalmente después de un número dado de iteraciones). En la fase de selección se escogen generalmente los individuos más aptos de la población actual para ser posteriormente recombinados en la fase de reproducción. Los individuos resultantes de la recombinación se alteran mediante un operador de mutación. Finalmente, a partir de la población actual y/o los mejores individuos generados (de acuerdo a su valor de *fitness*) se forma la nueva población, dando paso a la siguiente generación del algoritmo.

Algoritmos de estimación de la distribución (EDA)

Los *algoritmos de estimación de la distribución* o *Estimation of Distribution Algorithms* (EDAs) [187] muestran un comportamiento similar a los algoritmos evolutivos presentados en la sección anterior y, de hecho, muchos autores consideran los EDAs como otro tipo de EA. Los EDAs operan sobre una población de soluciones tentativas como los algoritmos evolutivos pero, a diferencia de estos últimos, que utilizan operadores de recombinación y mutación para mejorar las soluciones, los EDAs infieren la distribución de probabilidad del conjunto seleccionado y, a partir de ésta, generan nuevas soluciones que formarán parte de la población.

Los modelos gráficos probabilísticos son herramientas comúnmente usadas en el contexto de los EDAs para representar eficientemente la distribución de probabilidad. Algunos autores [145, 200, 229] han propuesto las redes bayesianas para representar la distribución de probabilidad en dominios discretos, mientras que las redes gaussianas se emplean usualmente en los dominios continuos [255].

Búsqueda dispersa (SS)

La *búsqueda dispersa* o *Scatter Search* (SS) [97] es una metaheurística cuyos principios fueron presentados en [95] y que actualmente está recibiendo una gran atención por parte de la comunidad científica [143]. El algoritmo se basa en mantener un conjunto relativamente pequeño de soluciones tentativas (llamado conjunto de referencia o *RefSet*) que se caracteriza por contener soluciones de calidad y diversas (distantes en el espacio de búsqueda). Para la definición completa de SS hay que concretar cinco componentes: creación de la población inicial, generación del conjunto de referencia, generación de subconjuntos de soluciones, método de combinación de soluciones y método de mejora. La Sección 7.2 incluye más detalles sobre esta técnica.

Optimización basada en colonias de hormigas (ACO)

Los *algoritmos de optimización basados en colonias de hormigas* o *Ant Colony Optimization* (ACO) [70, 71] están inspirados en el comportamiento de las hormigas reales cuando buscan comida. Este comportamiento es el siguiente: inicialmente, las hormigas exploran el área cercana a su nido de forma aleatoria. Tan pronto como una hormiga encuentra comida, la lleva al nido. Mientras que realiza este camino, la hormiga va depositando una sustancia química denominada feromona. Esta sustancia ayudará al resto de las hormigas a encontrar la comida. La comunicación indirecta entre las hormigas mediante el rastro de feromona las capacita para encontrar el camino más corto entre el nido y la comida. Este comportamiento es el que intenta simular este método 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. El algoritmo no fija ninguna planificación o sincronización *a priori* entre las fases, pudiendo ser incluso realizadas simultáneamente.

Optimización basada en cúmulos de partículas (PSO)

Los *algoritmos de optimización basados en cúmulos de partículas* o *Particle Swarm Optimization* (PSO) [133] están inspirados en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. El algoritmo PSO mantiene un conjunto de soluciones, también llamadas *partículas*, que son inicializadas aleatoriamente en el espacio de búsqueda. Cada partícula posee una posición y velocidad que cambia conforme avanza la búsqueda. En el movimiento de una partícula influye su velocidad y las posiciones donde la propia partícula y las demás de su vecindario encontraron buenas soluciones. En el contexto de PSO, el *vecindario de una partícula* se define como un conjunto de partículas del cúmulo. No debe confundirse con el concepto de vecindario de una solución utilizado previamente en este capítulo. El vecindario de una partícula puede ser *global*, en el cual todas las partículas del cúmulo se consideran vecinas, o *local*, en el que sólo las partículas más cercanas se consideran vecinas.

3.3. Metaheurísticas para optimización multiobjetivo

La mayoría de los problemas de optimización del mundo real son de naturaleza multiobjetivo, lo que supone que hay que minimizar o maximizar varias funciones a la vez que están normalmente en conflicto entre sí (problemas multiobjetivo o MOPs, *Multiobjective Optimization Problems*). Debido a la falta de soluciones metodológicas adecuadas, los problemas multiobjetivo se han resuelto en el pasado como problemas monoobjetivo. Sin embargo, existen diferencias fundamentales en los principios de funcionamiento de los algoritmos para optimización mono y multiobjetivo. Así, las técnicas utilizadas para resolver MOPs no se restringen normalmente a encontrar una solución única, sino un conjunto de soluciones de compromiso entre los múltiples objetivos contrapuestos, ya que no suele existir una solución que optimice simultáneamente todos los objetivos. Se pueden distinguir, por tanto, dos etapas cuando se aborda este tipo de problemas: por un lado, la optimización de varias funciones objetivo involucradas y, por otro, el proceso de toma de decisiones sobre qué solución de compromiso es la más adecuada [44]. Atendiendo a cómo manejan estas dos etapas, las técnicas para resolver MOPs se pueden clasificar en [45]:

- *A priori*: cuando las decisiones se toman antes de buscar soluciones.
- *Progresivas*: cuando se integran la búsqueda de soluciones y la toma de decisiones.
- *A posteriori*: cuando se busca antes de tomar decisiones.

Cada una de ellas tiene ciertas ventajas e inconvenientes que las hacen más adecuadas para determinados escenarios concretos [44, 65]. No obstante, en las dos primeras clases la búsqueda está muy influenciada por la decisión de un experto (*decision maker*) que determina la importancia de un objetivo sobre otro y que puede limitar arbitrariamente el espacio de búsqueda, impidiendo una resolución óptima del problema. En las técnicas *a posteriori*, por el contrario, se realiza una exploración lo más amplia posible para generar tantas soluciones de compromiso como sea posible. Es, entonces, cuando tiene lugar el proceso de toma de decisiones por parte del experto. Precisamente por la aproximación que siguen, estas técnicas *a posteriori* están siendo muy utilizadas dentro del campo de las metaheurísticas y, particularmente, en el campo de la computación evolutiva [44, 65]. Más concretamente, los algoritmos más avanzados aplican técnicas *a posteriori* basadas en el concepto de *Optimalidad de Pareto* [197] y es el enfoque seguido en esta tesis. Así, hemos estructurado esta sección en tres apartados. El primero de ellos presenta formalmente los conceptos básicos relacionados con esta optimalidad de Pareto. El siguiente apartado presenta las metas que debe perseguir todo algoritmo que utiliza estas técnicas cuando aborda un MOP. Finalmente, el tercer apartado discute algunos aspectos de diseño que se deben adoptar en los algoritmos que resuelven problemas siguiendo la aproximación anterior.

3.3.1. Conceptos básicos

En esta sección se presentan algunos conceptos básicos de la optimización multiobjetivo para familiarizar al lector con este campo. Comenzaremos dando unas nociones de lo que entendemos por problema de optimización multiobjetivo o MOP. Informalmente, un MOP se puede definir como el problema de encontrar un vector de variables de decisión que satisface un conjunto de restricciones y que optimiza un conjunto de funciones objetivo. Estas funciones forman una descripción matemática de criterios de rendimiento que están normalmente en conflicto entre sí. Por tanto, el término "optimización" se refiere a la búsqueda de una solución tal que contenga valores aceptables para todas las funciones objetivo [196].

Matemáticamente, la formulación de un MOP extiende la definición clásica de optimización monoobjetivo (Definición 1) para considerar la existencia de varias funciones objetivo. No hay, por tanto, una única solución al problema, sino un conjunto de soluciones. Este conjunto de soluciones se encuentra mediante la utilización de la Teoría de Optimalidad de Pareto [78]. Formalmente [242]:

Definición 6 (MOP). *Encontrar un vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ *que satisfaga las* m *restricciones de desigualdad* $g_i(\vec{x}) \geq 0, i = 1, 2, \ldots, m$*, las* p *restricciones de igualdad* $h_i(\vec{x}) = 0, i = 1, 2, \ldots, p$*, y que minimice la función vector* $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T$*, donde* $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ *es el vector de decisión de variables.*

El conjunto de todos los valores que satisfacen las restricciones define la *región de soluciones factibles* Ω y cualquier punto en $\vec{x} \in \Omega$ es una *solución factible.*

Teniendo varias funciones objetivo, la noción de "óptimo" cambia, ya que el objetivo para cualquier MOP es encontrar buenos compromisos (*trade-offs*) entre estas funciones. La noción de "óptimo" más utilizada es la propuesta por Francis Ysidro Edgeworth [77], generalizada posteriormente por Vilfredo Pareto [197]. Aunque algunos autores lo denominan óptimo de Edgeworth-Pareto, el término óptimo de Pareto es comúnmente aceptado. Su definición formal se da a continuación:

Definición 7 (Optimalidad de Pareto). *Un punto* $\vec{x}^* \in \Omega$ *es un óptimo de Pareto si para cada* $\vec{x} \in \Omega$ *y* $I = \{1, 2, \ldots, k\}$*, o bien* $\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*))$ *o bien hay al menos un* $i \in I \mid f_i(\vec{x}) > f_i(\vec{x}^*)$.

Esta definición dice que $\vec{x}^*$ es un óptimo de Pareto si no existe ningún vector factible $\vec{x}$ que mejore algún criterio sin causar simultáneamente un empeoramiento en al menos otro criterio

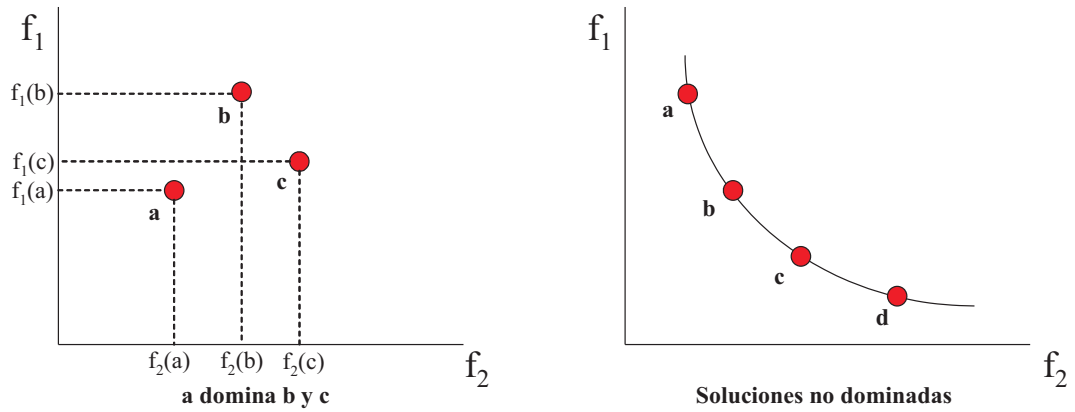


Figura 3.3: Ejemplo del concepto de dominancia de Pareto.

(asumiendo minimización). El concepto de optimalidad de Pareto es integral tanto a la teoría como a la resolución de MOPs. Existen un algunas definiciones adicionales que son también básicas en optimización multiobjetivo [242]:

Definición 8 (Dominancia de Pareto)**.** *Un vector* $\vec{u} = (u_1, \ldots, u_k)$ *se dice que domina a otro vector* $\vec{v}$=$(v_1, \ldots, v_k)$ *(representado por* $\vec{u} \prec \vec{v}$*) si y sólo si* $\vec{u}$ *es parcialmente menor que* $\vec{v}$*, es decir,* $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \ \wedge\ \exists\, i \in \{1, \ldots, k\} :\ u_i < v_i$.

Ilustremos gráficamente este concepto. La Figura 3.3 incluye dos conjuntos de soluciones para un problema multiobjetivo con dos funciones f_1 y f_2, que han de ser minimizadas. Siendo ambos objetivos igual de importantes, no resulta trivial distinguir qué solución es mejor que otra. Podemos utilizar la definición anterior para esto. Así, si nos fijamos en la parte izquierda de la figura, podemos decir que a es mejor que b puesto que $f_1(a) < f_1(b)$ y $f_2(a) < f_2(b)$, es decir, es mejor en ambos objetivos y, por tanto, se dice que a domina a b ($a \prec b$). Lo mismo ocurre si comparamos a y c, en ambos objetivos $f_1(a) < f_1(c)$ y $f_2(a) < f_2(c)$, por lo que $a \prec c$. Comparemos ahora las soluciones b y c entre ellas. Se puede observar que c es mejor que b en f_1 ($f_1(c) < f_1(b)$, pero b es mejor que c para f_2 ($f_2(b) < f_2(b)$). Según la Definición 8, no podemos decir que b domina a c ni que c domina a b, es decir, no podemos concluir que una solución es mejor que la otra. En este caso se dice que ambas soluciones son no dominadas. En la parte derecha de la Figura 3.3 se muestran 4 soluciones de este tipo, en las que ninguna es mejor que las demás.

Resolver un MOP consiste, por tanto, en encontrar el conjunto de soluciones que dominan a cualquier otra solución del espacio de soluciones, lo que significa que son las mejores para el problema y, por tanto, conforman su solución óptima. Formalmente:

Definición 9 (Conjunto Óptimo de Pareto)**.** *Para un MOP dado* $\vec{f}(\vec{x})$*, el conjunto óptimo de Pareto se define como* $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg\exists \vec{x'} \in \Omega, \vec{f}(\vec{x'}) \preccurlyeq \vec{f}(\vec{x})\}$.

No hay que olvidar que las soluciones Pareto-óptimas que están en $\mathcal{P}^*$ están en el espacio de las variables (genotípico), pero cuyos componentes del vector en el espacio de objetivos (fenotípico) no se pueden mejorar simultáneamente. Estas soluciones también suelen llamarse *no inferiores*, *admisibles* o *eficientes*. El frente de Pareto se define, entonces, como:

Definición 10 (Frente de Pareto)**.** *Para un MOP dado* $\vec{f}(\vec{x})$ *y su conjunto óptimo de Pareto* $\mathcal{P}^*$*, el frente de Pareto se define como* $\mathcal{PF}^* = \{\vec{f}(\vec{x}), \vec{x} \in \mathcal{P}^*\}$.

$$
\begin{array}{lcll}
\text{Min } F & = & (f_1(\vec{x}), f_2(\vec{x})) & \\
f_1(\vec{x}) & = & 4x_1^2 + 4x_2^2 & \\
f_2(\vec{x}) & = & (x_1 - 5)^2 + (x_2 - 5)^2 & \\
\text{Subject to:} & & & \\
g1(\vec{x}) & = & (x_1 - 5)^2 + x_2^2 - 25 & \leq 0 \\
g2(\vec{x}) & = & -(x_1 - 8)^2 - (x_2 + 3)^2 + 7{,}7 & \leqslant 0 \\
0 & \leq & x_1 & \leqslant 5 \\
0 & \leq & x_2 & \leqslant 3
\end{array}
$$

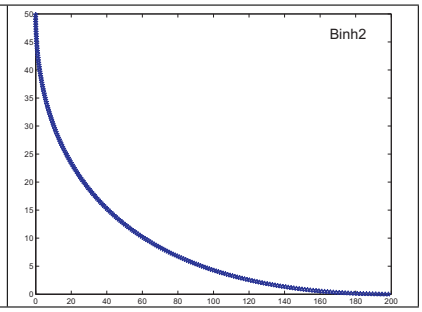


Figura 3.4: Formulación y frente de Pareto del problema Bihn2.

$$
\begin{array}{lcl}
f_1(x) & = & (1 + g(x_M)) \cos(x_1^{\alpha} \frac{\pi}{2}) \cos(x_2^{\alpha} \frac{\pi}{2}) \\
f_2(x) & = & (1 + g(x_M)) \cos(x_1^{\alpha} \frac{\pi}{2}) \sin(x_2^{\alpha} \frac{\pi}{2}) \\
f_3(x) & = & (1 + g(x_M)) \sin(x_1^{\alpha} \frac{\pi}{2}) \\
 & & 0 \leq x_i \leq 1, \quad i = 1, 2, \ldots, n \\
 & & \\
g(x_M) & = & \sum_{x_i \in x_M} (x_i - 0{,}5)^2 \\
 & & \\
n & = & 12 \\
x_M & = & x_3, ..., x_{12} \\
\alpha & = & 100
\end{array}
$$

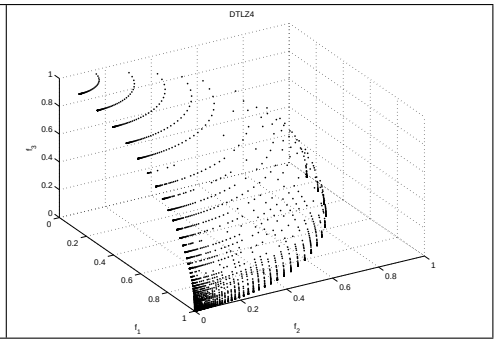


Figura 3.5: Formulación y frente de Pareto del problema DTLZ4.

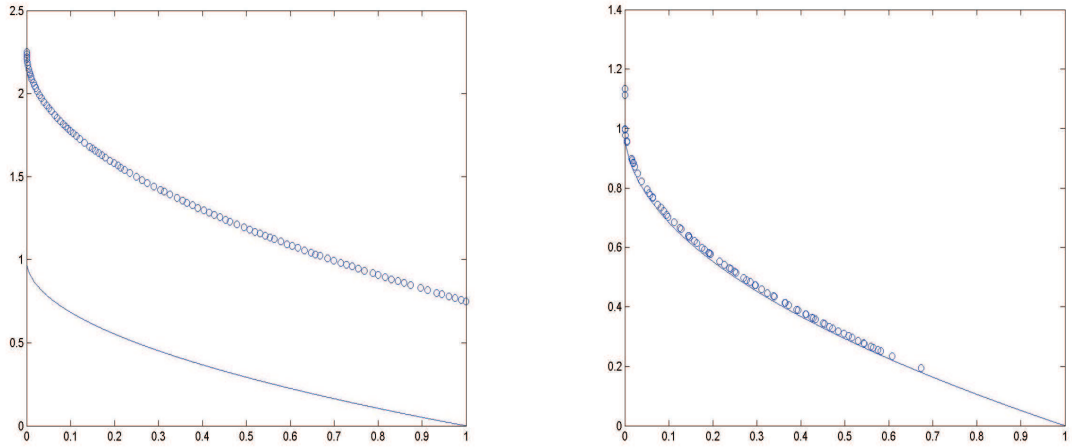
Es decir, el frente de Pareto está compuesto por los valores en el espacio de objetivos del conjunto óptimo de Pareto. En general, no es fácil encontrar una expresión analítica de la línea o superficie que contiene estos puntos. De hecho, en la mayoría de los casos es imposible. Como ejemplo, las Figuras 3.4 y 3.5 muestran la formulación y su correspondiente frente de Pareto de los problemas Binh2 y DTLZ4 [44]. En el primer caso se trata de un problema biobjetivo, f_1 y f_2, con dos variables de decisión x_1 y x_2, que tiene definidas, además, dos restricciones $g1$ y $g2$. El problema DTLZ4, por su parti, tiene tres objetivos y ninguna restricción (la función $g()$ aquí es únicamente una notación usada para su formulación).

3.3.2. Objetivos en la resolución de MOPs

Cuando se aborda la resolución de un problema de optimización multiobjetivo, la principal meta de todo algoritmo de optimización que utiliza los conceptos y técnicas descritas en la sección anterior es encontrar su frente de Pareto (o, lo que es lo mismo, su conjunto óptimo de Pareto). No obstante, la presencia de múltiples soluciones Pareto-óptimas hace que sea difícil elegir una solución sobre otra sin información adicional sobre el problema, ya que todas estas soluciones son igualmente importantes. Dado un MOP, por tanto, se busca idealmente un número de soluciones no dominadas que persiguen dos metas:

1. Encontrar un conjunto de soluciones lo más cercano posible al frente de Pareto óptimo.

2. Encontrar un conjunto de soluciones tan uniformemente diverso como sea posible.



(a) (b)

Figura 3.6: Ejemplos de mala convergencia (a) y diversidad (b) en frentes de Pareto.

Mientras que la primera meta, converger hacia la solución óptima, es obligatoria en toda tarea de optimización mono o multiobjetivo, la segunda es completamente específica para optimización multiobjetivo. Además de converger hacia el frente óptimo, las soluciones deben estar uniformemente repartidas a lo largo de todo el frente. Sólo con un conjunto diverso de soluciones se puede asegurar, por una parte, un buen conjunto de soluciones de compromiso entre los diferentes objetivos para la posterior toma de decisiones por parte del experto y, por otra, que se ha realizado una buena exploración del espacio de búsqueda. La Figura 3.6 muestra dos ejemplos de frentes que fallan, cada uno, en una de las metas anteriores. En la parte (a) podemos ver una aproximación al frente en el que las soluciones no dominadas se distribuyen perfectamente. No obstante, se trata de un MOP diseñado de forma que contiene múltiples frentes engañosos y, en realidad, las soluciones obtenidas no son Pareto-óptimas, aunque su diversidad es excelente. Por el contrario, en la parte (b) de la misma figura, tenemos un conjunto de soluciones que han convergido hacia el frente de Pareto óptimo pero, sin embargo, deja regiones de éste sin cubrir. Aunque ninguno de los dos casos es deseable, la primera situación es claramente peor: ninguna de las soluciones obtenidas es Pareto-óptima.

3.3.3. Aspectos de diseño

Adoptar técnicas basadas en optimalidad de Pareto dentro de algoritmos metaheurísticos supone, por un lado, trabajar con soluciones no dominadas que hacen necesario incorporar mecanismos específicos para manejarlas y, por otro, encontrar no una solución única, sino un conjunto de soluciones Pareto-óptimas que, además, ha de tener la diversidad suficiente para cubrir el todo el frente. Si bien existen muchos aspectos a considerar dependiendo de cada algoritmo concreto, los siguientes se pueden considerar comunes a todos ellos: función de aptitud (*función de fitness*) de las soluciones, mantenimiento de la diversidad, y manejo de restricciones. Cada uno de ellos se discute a continuación.

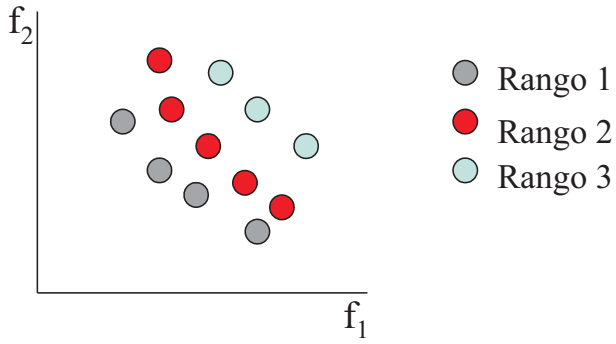


Figura 3.7: Ejemplo de ordenación (*ranking*) de soluciones en un MOP con dos objetivos.

Función de aptitud

En el ciclo de funcionamiento de toda metaheurística siempre existe alguna fase en la que hay que ordenar las soluciones con las que trabaja según su función de fitness para seleccionar alguna de ellas. Hablamos, por ejemplo, de los operadores de selección y reemplazo en algoritmos evolutivos o el método de actualización del conjunto de referencia en búsqueda dispersa. En el caso de optimización monoobjetivo, el fitness de una solución es un valor único y la ordenación de soluciones es trivial de acuerdo con este valor. No obstante, en nuestra aproximación para resolver MOPs, el fitness es un vector de valores (un valor por cada objetivo) por lo que la ordenación no es tan directa.

La relación de dominancia (Ecuación 8) es la clave en este tipo de técnicas basadas en optimalidad de Pareto, ya que nos va a permitir establecer una ordenación de las soluciones. De hecho, esta relación es una relación de orden parcial estricto, puesto que no es reflexiva, ni simétrica, ni antisimétrica, pero sí transitiva. Así, se han propuesto diferentes métodos en la literatura [44, 65] que, básicamente, transforman el vector de fitness en un valor único utilizando esta relación. Esta estrategia fue originalmente propuesta por Goldberg en [101] para guiar la población de un GA hacia el frente de Pareto de un MOP. La idea básica consiste en encontrar las soluciones de la población que no están dominadas por ninguna otra. A estas soluciones se le asigna el mayor orden (las mejores en la ordenación establecida por la relación de dominancia). A continuación, se consideran las soluciones no dominadas que quedan si se eliminan todas las anteriores, a las que se asigna el siguiente rango. El proceso continúa hasta que se le asigna un rango a todas las soluciones. La Figura 3.7 muestra un ejemplo del funcionamiento de este método de ordenación (f_1 y f_2 son funciones que han de minimizarse). Esta ordenación basada en dominancia es la más básica. Otra más avanzadas, como la fuerza o *strength* de SPEA2 [262] tienen en cuenta, además, el número de soluciones a los que domina cada solución.

Diversidad

Si bien la función de fitness basada en dominancia ya dirige la búsqueda hacia el frente de Pareto dando una mayor aptitud a las soluciones no dominadas, esta aproximación por sí sola no es suficiente cuando se aborda un MOP. Si recordamos la Sección 3.3.2, además de converger al frente óptimo, las soluciones han de distribuirse lo mejor posible sobre este frente para poder ofrecer al experto el abanico más amplio de soluciones al problema multiobjetivo.

Aunque existen diferentes aproximaciones en la literatura [44], las más utilizadas en los algoritmos del estado del arte están basadas en complementar la función de fitness basada en dominancia (sección anterior) con un estimador que mide la densidad, en el espacio de objetivos, de soluciones

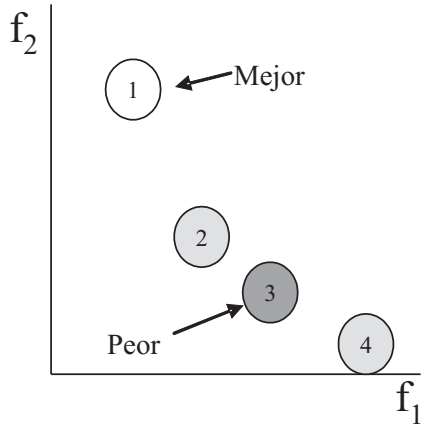


Figura 3.8: Ejemplo de estimador de densidad para soluciones no dominadas en un MOP con dos objetivos.

alrededor de una solución dada. Así, dada dos soluciones con el mismo fitness (*ranking*, *strength*), el estimador de densidad discrimina entre las mejores y peores soluciones atendiendo a la diversidad de las mismas. Consideremos, por ejemplo, el conjunto de soluciones no dominadas de la Figura 3.8. Según la densidad de las mismas, la solución 1 se puede considerar como la mejor puesto es la que está en la zona menos "poblada". La solución 3, por el contrario, sería la peor al estar en una zona del frente donde ya existen soluciones cercanas. Algunos de los estimadores de densidad propuestos por los algoritmos multiobjetivo más conocidos son: *niching* en MOGA [86] y NSGA [230], el grid adaptativo de PAES [136], *crowding* en NSGA-II [68] y la distancia al k-ésimo vecino de SPEA2 [262].

Manejo de restricciones

La definición de problema multiobjetivo (Ecuación 6) incluida en la Sección 3.3.1 incluye explícitamente restricciones ya que, principalmente, es la situación típica cuando se consideran problemas de corte real, como los considerados en esta tesis. Las restricciones se pueden considerar como duras o débiles. Una restricción es dura cuando ha de satisfacerse para que una solución dada sea aceptable. Por el contrario, una restricción débil es aquella que se puede relajar de alguna forma para aceptar una solución.

La aproximación más utilizada en las metaheurísticas multiobjetivo del estado del arte para tratar con restricciones están basadas en un esquema en el que las soluciones factibles son superiores a las no factibles [64, 65]. Así, dadas dos soluciones que se han de comparar, se pueden dar tres casos:

1. Si ambas soluciones son factibles, se utiliza la función de fitness basada en dominancia de Pareto explicada en la Sección 3.3.3. En el caso de que ambas sean no dominadas (mismo fitness), se utiliza un estimador de densidad (Sección 3.3.3) para discriminar entre ellas.

2. Si una solución es factible y la otra no, la factible se considera como mejor.

3. Si ambas soluciones son no factibles, entonces se selecciona la que menos viola las restricciones.

Queda por determinar cómo se cuantifica la cantidad de violación de restricciones de una solución dada. Para esto, la estrategia más utilizada consiste en transformar todas las restricciones para que sean de tipo *mayor-o-igual-que* cero: $g_i(\vec{x}) \geq 0$, según la definición de MOP (Ecuación 6) [65].

Se puede considerar como un tipo de normalización, de forma que el propio valor $g_i(\vec{x})$ se usa para medir cuánto se viola la restricción. El mayor inconveniente para esta estrategia viene dado por la restricciones de igualdad $h_i(\vec{x}) = 0$. Si se trata de una restricción débil, se puede relajar directamente a $h_i(\vec{x}) \geq 0$. Sin embargo, si $h_i(\vec{x}) = 0$ es una restricción dura, la transformación no es directa (especialmente cuando es una restricción no lineal). Según un resultado obtenido en [63], es posible convertir estas restricciones duras de igualdad en restricciones débiles con pérdidas de precisión, lo que permite considerar ya todas las restricciones del mismo tipo. Existen otras muchas estrategias para tratar con restricciones en optimización multiobjetivo [44, 65] pero sólo hemos detallado la que se utilizará en esta tesis.

3.4. Metaheurísticas paralelas

Aunque el uso de metaheurísticas permite reducir significativamente la complejidad temporal del proceso de búsqueda, este tiempo puede seguir siendo muy elevado en algunos problemas de interés real. Con la proliferación de plataformas paralelas de cómputo eficientes, la implementación paralela de estas metaheurísticas surge de forma natural como una alternativa para acelerar la obtención de soluciones precisas a estos problemas. La literatura es muy extensa en cuanto a la paralelización de metaheurísticas se refiere [8, 52, 59, 163] ya que se trata de una aproximación que puede ayudar no sólo a reducir el tiempo de cómputo, sino a producir también una mejora en la calidad de las soluciones encontradas. Esta mejora está basada en un nuevo modelo de búsqueda que alcanza un mejor balance entre intensificación y diversificación. De hecho, muchos investigadores no utilizan plataformas paralelas de cómputo para ejecutar estos modelos paralelos y, aún así, siguen obteniendo mejores resultados que con los algoritmos secuenciales tradicionales. Tanto para las metaheurísticas basadas en trayectoria como para las basadas en población se han propuesto modelos paralelos acorde a sus características. Las siguientes secciones 3.4.1 y 3.4.2 presentan, respectivamente, generalidades relacionadas con la paralelización de cada tipo de metaheurística.

En nuestro contexto, donde vamos a considerar problemas del mundo real, la utilización no sólo de modelos paralelos, sino también de plataformas paralelas es casi obligatorio, ya que una simple evaluación del problema puede llevar minutos o incluso horas si se consideran problemas en los que intervienen, por ejemplo, complejas simulaciones. En estos últimos casos, ni siquiera las arquitecturas paralelas más usuales como los clusters de máquinas o los multiprocesadores de memoria compartida tienen la capacidad suficiente para abordar, en un tiempo razonable, estos problemas de optimización que implican tareas tan costosas computacionalmente. La aparición de los sistemas de computación grid [26, 90] puede permitir solventar estos inconvenientes en cierta medida, ya que son capaces de agrupar, como si de un elemento único se tratase, la potencia computacional de una gran cantidad de recursos geográficamente distribuidos. No obstante, todos los modelos paralelos de metaheurísticas no se adecúan a este tipo de plataformas y explotan toda su capacidad de cómputo. La Sección 3.4.3 está dedicada a definir en detalle en qué consiste un sistema de computación grid, así como a la introducción de aspectos de diseño claves para el despliegue de metaheurísticas en este tipo de plataformas de cómputo.

3.4.1. Modelos paralelos para métodos basados en trayectoria

Los modelos paralelos de metaheurísticas basadas en trayectoria encontrados en la literatura se pueden clasificar, generalmente, dentro de tres posibles esquemas: ejecución en paralelo de varios métodos (*modelo de múltiples ejecuciones*), exploración en paralelo del vecindario (*modelo de movimientos paralelos*), y cálculo en paralelo de la función de *fitness* (*modelo de aceleración del movimiento*). A continuación detallamos cada uno de ellos.

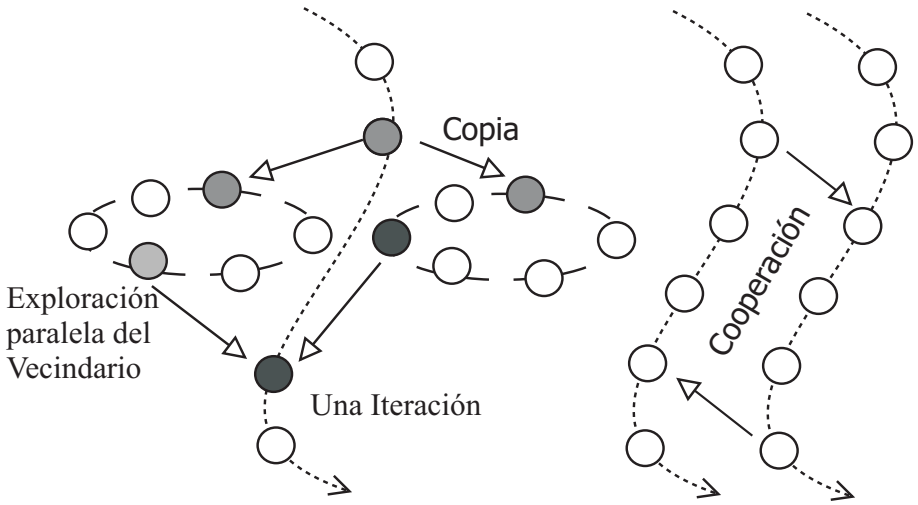


Figura 3.9: Modelos paralelos más usados en los métodos basados en trayectoria. A la izquierda se muestra el modelo de movimientos paralelos, donde se hace una exploración paralela del vecindario. A la derecha, se detalla el modelo de ejecuciones múltiples con cooperación, donde hay varios métodos ejecutándose en paralelo y cooperando entre ellos.

- **Modelo de múltiples ejecuciones:** este modelo consiste en ejecutar en paralelo varios subalgoritmos ya sean homogéneos o heterogéneos [157, 12]. En general, cada subalgoritmo comienza con una solución inicial diferente. Se pueden distinguir diferentes casos dependiendo de si los subalgoritmos colaboran entre sí o no. El caso en el que las ejecuciones son totalmente independientes se usa ampliamente porque es simple de utilizar y muy natural. En este caso, la semántica del modelo es la misma que la de la ejecución secuencial, ya que no existe cooperación. El único beneficio al utilizar este modelo respecto a realizar las ejecuciones en una única máquina es la reducción del tiempo de ejecución total.

  Por otro lado, en el caso cooperativo (véase el ejemplo de la derecha de la Figura 3.9), los diferentes subalgoritmos intercambian información durante la ejecución. En este caso el comportamiento global del algoritmo paralelo es diferente al secuencial y su rendimiento se ve afectado por cómo esté configurado este intercambio. El usuario debe fijar ciertos parámetros para completar el modelo: qué información se intercambian, cada cuánto se pasan la información y cómo se realiza este intercambio. La información intercambiada suele ser la mejor solución encontrada, los movimientos realizados o algún tipo de información sobre la trayectoria realizada. En cualquier caso, esta información no debe ser abundante para que el coste de la comunicación no sea excesivo e influya negativamente en la eficiencia. También se debe fijar cada cuántos pasos del algoritmo se intercambia la información. Para elegir este valor hay que tener en cuenta que el intercambio no sea muy frecuente, para que el coste de la comunicación no sea perjudicial, ni muy poco frecuente, para que el intercambio tenga algún efecto en el comportamiento global. Por último, se debe indicar si las comunicaciones se realizarán de forma asíncrona o síncrona. En el caso síncrono, los subalgoritmos, cuando llegan a la fase de comunicación, se detienen hasta que todos ellos llegan a este paso y sólo entonces se realiza la comunicación. En el caso asíncrono, que es el más usado, cada subalgoritmo realiza el intercambio sin esperar al resto cuando llega al paso de comunicación.

- **Modelo de movimientos paralelos:** los métodos basados en trayectoria en cada paso examinan parte de su vecindario y, de él, eligen la siguiente solución a considerar. Este

paso suele ser computacionalmente costoso, ya que examinar el vecindario implica múltiples cálculos de la función de *fitness*. El modelo de movimientos paralelos tiene como objetivo acelerar dicho proceso mediante la exploración en paralelo del vecindario (véase el esquema de la izquierda de la Figura 3.9). Siguiendo un modelo maestro-esclavo, el maestro (el que ejecuta el algoritmo) pasa a cada esclavo la solución actual. Cada esclavo explora parte del vecindario de esta solución devolviendo la más prometedora. Entre todas estas soluciones devueltas el maestro elige una para continuar el proceso. Este modelo no cambia la semántica del algoritmo, sino que simplemente acelera su ejecución en caso de ser lanzado en una plataforma paralela. Este modelo es bastante popular debido a su simplicidad.

- **Modelo de aceleración del movimiento:** en muchos casos, el proceso más costoso del algoritmo es el cálculo de la función de *fitness*. Este cálculo, en muchos problemas, se puede descomponer en varios cómputos independientes más simples que, una vez llevados a cabo, se pueden combinar para obtener el valor final de la función de *fitness*. En este modelo, cada uno de esos cálculos más simples se asignan a los diferentes procesadores y se realizan en paralelo, acelerando el cálculo total. Al igual que el anterior, este modelo tampoco modifica la semántica del algoritmo respecto a su ejecución secuencial.

3.4.2. Modelos paralelos para métodos basados en población

El paralelismo surge de manera natural cuando se trabaja con poblaciones, ya que cada individuo puede manejarse de forma independiente. Debido a esto, el rendimiento de los algoritmos basados en población suele mejorar bastante cuando se ejecutan en paralelo. A alto nivel podemos dividir las estrategias de paralelización de este tipo de métodos en dos categorías: (1) paralelización del cómputo, donde las operaciones que se llevan a cabo sobre los individuos son ejecutadas en paralelo, y (2) paralelización de la población, donde se procede a la estructuración de la población.

Uno de los modelos más utilizado que sigue la primera de las estrategias es el denominado *maestro-esclavo* (también conocido como *paralelización global*). En este esquema, un proceso central realiza las operaciones que afectan a toda la población (como, por ejemplo, la selección en los algoritmos evolutivos) mientras que los procesos esclavos se encargan de las operaciones que afectan a los individuos independientemente (como la evaluación de la función de *fitness*, la mutación e incluso, en algunos casos, la recombinación). Con este modelo, la semántica del algoritmo paralelo no cambia respecto al secuencial pero el tiempo global de cómputo es reducido. Este tipo de estrategias son muy utilizadas en las situaciones donde el cálculo de la función de *fitness* es un proceso muy costoso en tiempo. Otra estrategia muy popular es la de acelerar el cómputo mediante la realización de múltiples ejecuciones independientes (sin ninguna interacción entre ellas) usando múltiples máquinas para, finalmente, quedarse con la mejor solución encontrada entre todas las ejecuciones. Al igual que ocurría con el modelo de múltiples ejecuciones sin cooperación de las metaheurísticas paralelas basadas en trayectoria, este esquema no cambia el comportamiento del algoritmo, pero permite reducir de forma importante el tiempo total de cómputo.

Al margen del modelo maestro-esclavo, la mayoría de los algoritmos paralelos basados en población encontrados en la literatura utilizan alguna clase de estructuración de los individuos de la población. Este esquema es ampliamente utilizado especialmente en el campo de los algoritmos evolutivos y es el que mejor ilustra esta categorización. Entre los esquemas más populares para estructurar la población encontramos el modelo *distribuido* (o de grano grueso) y el modelo *celular* (o de grano fino) [14].

En el caso de los algoritmos distribuidos [6] (véase el esquema de la derecha en la Figura 3.10), la población se divide entre un conjunto de islas que ejecutan una metaheurística secuencial. Las islas cooperan entre sí mediante el intercambio de información (generalmente individuos, aunque nada impide intercambiar otro tipo de información). Esta cooperación permite introducir diversidad en

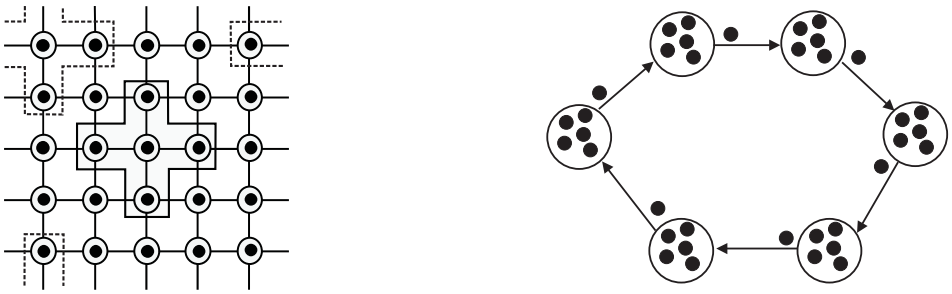


Figura 3.10: Los dos modelos más populares para estructurar la población: a la izquierda el modelo celular y a la derecha el modelo distribuido.

las subpoblaciones, evitando caer así en los óptimos locales. Para terminar de definir este esquema el usuario debe dar una serie de parámetros como: la topología, que indica a dónde se envían los individuos de cada isla y de dónde se pueden recibir; el periodo de migración, que es el número de iteraciones entre dos intercambios de información; la tasa de migración, que es el número de individuos emigrados; el criterio de selección de los individuos a migrar y criterio de reemplazo, que indica si se reemplazan algunos individuos de la población actual para introducir a los inmigrantes y determina qué individuos se reemplazarán. Finalmente, se debe decidir si estos intercambios se realizan de forma síncrona o asíncrona.

Por otro lado, las metaheurísticas celulares [75] (véase el esquema de la izquierda en la Figura 3.10) se basan en el concepto de vecindario[1]. Cada individuo tiene a su alrededor un conjunto de individuos vecinos donde se lleva a cabo la explotación de las soluciones. La exploración y la difusión de las soluciones al resto de la población se produce debido a que los vecindarios están solapados, lo que produce que las buenas soluciones se extiendan lentamente por toda la población.

A parte de estos modelos básicos, en la literatura también se han propuesto modelos híbridos donde se implementan esquemas de dos niveles. Por ejemplo, una estrategia bastante común en la literatura es aquélla donde en el nivel más alto tenemos un esquema de grano grueso, mientras que cada subpoblación se organiza siguiendo un esquema celular.

3.4.3. Utilización de sistemas de computación grid

Nuestra propuesta en esta tesis consiste en diseñar modelos paralelos eficientes para metaheurísticas de forma que podamos abordar problemas del mundo real utilizando plataformas grid. Las dos secciones siguientes se dedican, la primera, a introducir el concepto de sistema de computación grid y, la segunda, a discutir algunos aspectos de diseño que se han de considerar en una metaheurística para utilizar este tipo de sistemas.

Definición de sistema grid

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. [166]. Sin embargo, el concepto "grid" aún no existía y se utilizaban entonces términos como metasistema o metacomputador (*metasystem* or *metacomputing*) para referirse a "los recursos computacionales disponibles de forma transparente al usuario mediante redes" [227]. Basándose en

[1] De nuevo aquí, la palabra *vecindario* se usa en el sentido de definir un conjunto de individuos que serán vecinos de uno dado, como en PSO. No debe confundirse con el concepto de vecindario en el espacio de soluciones que se usa en la búsqueda local o en metaheurísticas como SA o VNS.

una analogía con la red eléctrica (en inglés, *power grid*), en la que se proporciona acceso consistente, fiable y transparente a la electricidad, independientemente de dónde se genera, a mediados de los 90 se acuñó el término "grid" para referirse a la infraestructura hardware y software que proporciona este tipo de acceso a recursos computacionales, con independencia de la distribución geográfica tanto de recursos como de usuarios [90].

Si bien no existe una definición clara y completa de grid [231], sí hay una lista de "mínimos" ampliamente aceptada [87]:

1. Un conjunto coordinado de recursos que no están sujetos a un control centralizado,
2. que utilizan protocolos e interfaces abiertos, estándares y de propósito general,
3. para proporcionar, de forma no trivial, diferentes calidades de servicios.

En realidad, la idea original de grid ha ido mucho más allá de lo inicialmente planeado. Un grid permite compartir, seleccionar y agregar no sólo recursos computacionales, sino también sistemas de almacenamiento, bases de datos o dispositivos especializados con el objetivo de resolver problemas de ciencia, ingeniería o comercio [22]. Desde el punto de vista de la construcción de grids hay que tener en cuenta un conjunto de características inherentes a este sistema como el elevado número de recursos, la distribución geográfica de los mismos, heterogeneidad, multiples dominios de administración, seguridad, coordinación entre recursos y acceso transparente, fiable y consistente [26]. Existen múltiples líneas de investigación que trabajan en este sentido y que han proporcionado ya herramientas como Globus [88, 89], Condor [238] y Proactive [21], que proporcionan la infraestructura software necesaria para construir este tipo de sistemas.

Metaheurísticas y sistemas grid

Nuestro aproximación en esta tesis trata de diseñar modelos paralelos de metaheurísticas [162, 164] que puedan aprovechar la enorme capacidad de cómputo que los sistemas grid son capaces de proporcionar. El objetivo está claro: resolver problemas reales en tiempos razonables.

Como se ha mencionado antes, un sistema grid se puede considerar como una colección de recursos computacionales distribuidos que están conectados mediante una red. En este contexto, podemos distinguir dos niveles diferentes de software. En el nivel superior, tenemos las aplicaciones grid que se ejecutan sobre el sistema; a un nivel inferior aparece el software del sistema grid, que es encargado de manejar la infraestructura grid y de permitir el desarrollo de aplicaciones grid.

Los recursos de un sistema grid comparte típicamente alguna de las siguientes características [26]:

1. Son numerosos,
2. están gestionados por diferentes organizaciones e individuos,
3. pueden desaparecer del sistema sin previo aviso,
4. se pueden añadir de forma dinámica al sistema,
5. tienen diferentes requisitos y políticas de seguridad,
6. son heterogéneos,
7. están conectados por redes heterogéneas,
8. tienen diferentes políticas de gestión de recursos, y
9. están, generalmente, geográficamente distribuidos.

Todos estos aspectos se deben manejar hasta cierto por el software del sistema grid, mientras que sólo algunas de ellas deben ser consideradas a nivel de aplicación grid. A continuación analizamos los elementos que pueden influenciar el diseño de una metaheurística para su ejecución sobre un sistema grid.

El hecho de que los recursos son numerosos (1) es el leitmotif de los sistemas grid, y la principal razón por la que se descarta el uso de modelos paralelos con topologías como anillos, mallas, etc. Por una parte, debido a que los recursos pueden desaparecer del sistema sin previo aviso (3) y a que se puede incorporar nuevos recursos al sistema (4), este tipo de topologías regulares son muy difíciles de implementar (i.e., ya que esta topología se debería reconfigurar en tiempo de ejecución). Por otra parte, los beneficios del intercambio de soluciones entre los distintos componentes del algoritmo paralelo puede ser complicado de conseguir en un sistema compuesto por miles de nodos (e.g., el efecto de migrar un individuo utilizando un anillo unidireccional puede no afectar a poblaciones que estén muy distantes en el anillo). Estas razones son las que nos han llevado a considerar paralelizaciones de metaheurísticas basadas en un modelo maestro/esclavo.

Este modelo paralelo ofrece varias ventajas. En primer lugar, se trata de un modelo conceptualmente simple: el maestro envía las tareas que involucran la evaluación de las soluciones a los esclavos, que responden con el valor de fitness de estas soluciones. En segundo lugar, se necesita una topología en estrella, que es fácil de implementar en un sistema grid. Finalmente, debido a la naturaleza estocástica de las metaheurísticas, el principio de funcionamiento del algoritmo no se ve afectado por la pérdida potencial de un esclavo (y la solución que está evaluando en ese momento).

3.5. Evaluación estadística de resultados

Como ya se ha comentado en varias ocasiones a lo largo de este documento, las metaheurísticas son técnicas no deterministas. Esto implica que diferentes ejecuciones del mismo algoritmo sobre un problema dado no tienen por qué encontrar la misma solución. Esta propiedad característica de las metaheurísticas supone un problema importante para los investigadores a la hora de evaluar sus resultados y, por tanto, a la hora de comparar su algoritmo con otros algoritmos existentes.

Existen algunos trabajos que abordan el análisis teórico para un gran número de heurísticas y problemas [102, 132], pero dada la dificultad que entraña este tipo de análisis teórico, tradicionalmente se analiza el comportamiento de los algoritmos mediante comparaciones empíricas. Para ello es necesario definir indicadores que permitan estas comparaciones. Podemos encontrar, en general, dos tipos diferentes de indicadores. Por un lado, tenemos aquellos que miden la calidad de las soluciones obtenidas. Dado que a lo largo del desarrollo de esta tesis se han abordado problemas de optimización tanto mono como multiobjetivo, hay que considerar indicadores de calidad diferentes para cada tipo ya que, si bien el resultado en el primer caso es una única solución (el óptimo global), en el segundo caso tenemos un conjunto de soluciones, el conjunto óptimo de Pareto (Ecuación 9). Por otro lado, están los indicadores que miden el rendimiento de los algoritmos y que hacen referencia a los tiempos de ejecución o a la cantidad de recursos computacionales utilizados. Aunque la discusión que incluimos en las secciones siguientes trata los dos tipos de indicadores por separado, ambos están íntimamente ligados y suelen utilizarse conjuntamente para la evaluación de metaheurísticas, ya que el objetivo de este tipo de algoritmos es encontrar soluciones de alta calidad en un tiempo razonable.

Una vez definidos los indicadores, hay que realizar un mínimo de ejecuciones independientes del algoritmo para obtener resultados estadísticamente consistentes. Un valor de 30 suele considerarse el mínimo aceptable, aunque valores de 100 son recomendables. No vale la mera inclusión de medias y desviaciones típicas (algo usual, e incorrecto, en la literatura), ya que se pueden obtener conclusiones erróneas. Así, es necesario realizar un análisis estadístico global para asegurar que estas conclusiones son significativas y no son provocadas por variaciones aleatorias. Este tema se aborda con más detenimiento en la Sección 3.5.3.

3.5.1. Indicadores de calidad

Estos indicadores son los más importantes a la hora de evaluar una metaheurística. Son distintos dependiendo de si se conoce o no la solución óptima del problema en cuestión (algo común para problema clásicos de la literatura, pero poco usual en problemas del mundo real). Como ya se ha mencionado anteriormente, hay que distinguir además entre indicadores para problemas monoobjetivo y multiobjetivo.

Indicadores para optimización monoobjetivo

Para instancias de problemas donde la solución óptima es conocida, es fácil definir un indicador para controlar la calidad de la metaheurística: el número de veces que aquella se alcanza (*hit rate*). Esta medida generalmente se define como el porcentaje de veces que se alcanza la solución óptima respecto al total de ejecuciones realizadas. Desgraciadamente, conocer la solución óptima no es un caso habitual para problemas realistas o, aunque se conozca, su cálculo puede ser tan pesado computacionalmente que lo que interesa es encontrar una buena aproximación en un tiempo menor. De hecho, es común que los experimentos con metaheurísticas estén limitados a realizar a lo sumo un esfuerzo computacional definido de antemano (visitar un máximo número de puntos del espacio de búsqueda o un tiempo máximo de ejecución).

En estos casos, cuando el óptimo no es conocido, se suelen usar medidas estadísticas del indicador correspondiente. Las más populares son la media y la mediana del mejor valor de fitness encontrado en cada ejecución independiente. En general, es necesario ofrecer otros datos estadísticos como la varianza o la desviación estándar, además del correspondiente análisis estadístico, para dar confianza estadística a los resultados.

En los problemas donde el óptimo es conocido se pueden usar ambas métricas, tanto el número de éxitos como la media/mediana del fitness final (o también del esfuerzo). Es más, al usar ambas se obtiene más información: por ejemplo, un bajo número de éxitos pero una alta precisión indica que raramente encuentra el óptimo pero que es un método robusto.

Indicadores para optimización multiobjetivo

Si bien el procedimiento para medir la calidad de las soluciones en problemas monoobjetivo está clara, dentro del campo multiobjetivo esto es un tema de investigación muy activo [137, 264], ya que el resultado de estos algoritmos es un conjunto de soluciones no dominadas y no una solución única. Hay que definir, por tanto, indicadores de calidad para aproximaciones al frente de Pareto. Hay normalmente dos aspectos a considerar para medir la calidad de un frente: convergencia y diversidad. La primera hace referencia a la distancia existente entre la aproximación y el frente de Pareto óptimo del problema, mientras que la segunda mide la uniformidad de la distribución de soluciones sobre el frente. Como ocurre para el caso monoobjetivo, existen indicadores atendiendo a si se conoce o no el frente óptimo. A continuación se muestran los indicadores de calidad utilizados en esta tesis (el lector interesado puede ver [44, 65] para otros indicadores de calidad definidos en la literatura):

- **Número de óptimos de Pareto.** En la resolución de algunos problemas multiobjetivo muy complejos, encontrar un número alto de soluciones no dominadas puede ser una tarea realmente dura para el algoritmo. En este sentido, el número de óptimos de Pareto encontrados se puede utilizar como una medida de la capacidad del algoritmo para explorar los espacios de búsqueda definidos por el problema multiobjetivo.

- **Cubrimiento de conjuntos – C(A, B).** El *cubrimiento de conjuntos* $C(A, B)$ calcula la proporción de soluciones en el conjunto B que son dominadas por soluciones del conjunto A:

$$C(A,B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|} \quad . \tag{3.18}$$

  Un valor de la métrica $C(A, B) = 1$ significa que todos los miembros de B son dominados por A, mientras que $C(A, B) = 0$ significa que ningún miembro de B es dominado por A. De esta forma, cuanto mayor sea $C(A, B)$, mejor es el frente de Pareto A con respecto a B. Ya que el operador de dominancia no es simétrico, $C(A, B)$ no es necesariamente igual a $1 - C(B, A)$, y tanto $C(A, B)$ como $C(B, A)$ deben ser calculados para entender cuántas soluciones de A son cubiertas por B y viceversa.

- **Distancia generacional – GD.** Esta métrica fue introducida por Van Veldhuizen y Lamont [243] para medir cómo de lejos están los elementos del conjunto de soluciones no dominadas encontradas respecto del conjunto óptimo de Pareto. Se define como:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad , \tag{3.19}$$

  donde n es el número de soluciones no dominadas, y d_i es la distancia Euclídea (medida en el espacio objetivo) entre cada una estas soluciones y el miembro más cercano del conjunto óptimo de Pareto. Según esta definición, esta claro que un valor de $GD = 0$ significa que todos los elementos generados están en el conjunto óptimo de Pareto.

- **Dispersión – Δ.** La *dispersión* o Δ [68] es un indicador de diversidad que mide la distribución de las soluciones obtenidas. Esta métrica se define como:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \quad , \tag{3.20}$$

  donde d_i es la distancia Euclídea entre dos soluciones consecutivas, $\bar{d}$ es la media de estas distancias, y d_f y d_l son las distancias euclídeas a las soluciones *extremas* (límite) del frente óptimo en el espacio objetivo (para más detalles, consulte [68]). Esta medida toma el valor cero para una distribución ideal, cuando hay una dispersión perfecta de las soluciones del frente de Pareto.

- **Hipervolumen – HV.** La métrica *hipervolumen* [263] es un indicador combinado de convergencia y diversidad que calcula el volumen, en el espacio de objetivos, cubierto por los miembros de un conjunto Q de soluciones no dominadas (la región acotada por la línea discontinua en la Figura 3.11, $Q = \{A, B, C\}$) para problemas en los que todos los objetivos deben ser minimizados. Matemáticamente, para cada solución $i \in Q$, un hipercubo v_i se construye utilizando un punto de referencia W (que puede estar compuesto por la peor solución para cada objetivo, por ejemplo) y la solución i como las esquinas de la diagonal del hipercubo. El punto de referencia se puede obtener simplemente construyendo un vector de los peores valores para las funciones. Así, HV se calcula como el volumen de la unión de todos los hipercubos:

$$HV = volumen\left(\bigcup_{i=1}^{|Q|} v_i\right) \quad . \tag{3.21}$$

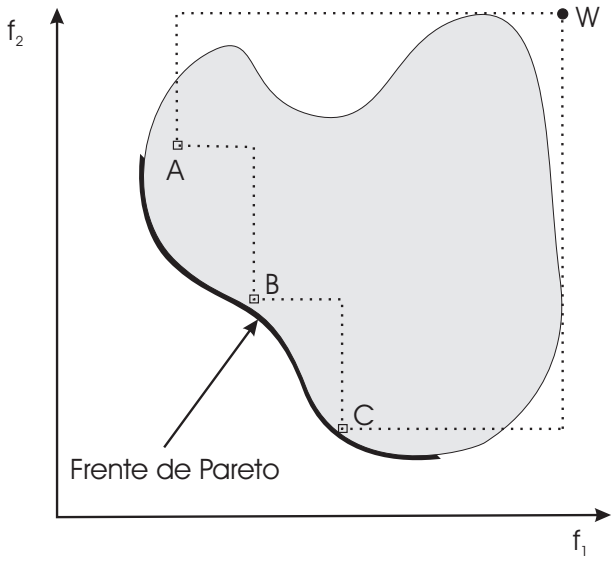


Figura 3.11: El hipervolumen cubierto por las soluciones no dominadas.

Hay que indicar, finalmente, que para las tres últimas métricas presentadas es necesario normalizar de alguna forma los frentes obtenidos para no obtener resultados engañosos. Para el caso de GD y Δ el proceso está claro: se normaliza respecto al frente óptimo ya que hay que disponer de él obligatoriamente para calcular el valor de estos indicadores de calidad. Calcular HV no requiere frente de referencia alguno, lo que permite medir calidad de frentes sin necesidad de conocer el conjunto óptimo de Pareto. No obstante, esta métrica es muy dependiente de la escala de los objetivos y la normalización es casi imprescindible para conseguir valores fiables de este indicador. Así, si se dispone del frente óptimo, se normaliza respecto a éste. Si, por el contrario, el frente óptimo no es conocido, cuando se quieren comparar varios algoritmos para el problema, el procedimiento que seguimos consiste en buscar los valores máximos y mínimos de todos los frentes del problema concreto en todos los algoritmos y normalizar respecto a esos valores. Así garantizamos que el punto de referencia para calcular los volúmenes v_i de HV es el mismo en todos los casos.

3.5.2. Indicadores de rendimiento

Entendemos por medidas de rendimiento aquellas que hacen referencia al tiempo o la cantidad de recursos computacionales utilizados por las metaheurísticas, que se suelen medir en base al número de soluciones visitadas del espacio de búsqueda (esfuerzo computacional) o al tiempo de ejecución.

Muchos investigadores prefieren el número de evaluaciones como manera de medir el esfuerzo computacional, ya que elimina los efectos particulares de la implementación, del software y del hardware, haciendo así que las comparaciones sean independientes de esos factores. Pero esta medida puede ser engañosa en algunos casos, ya que puede ocurrir que algunas evaluaciones tarden más que otras (algo muy común en programación genética [139]) o incluso que los operadores que manipulan las soluciones sean más costosos en una técnica u otra. En general, es recomendable usar las dos métricas (evaluaciones y tiempo) para obtener una medida realista del esfuerzo computacional.

Dado que vamos a evaluar algoritmos que se pueden ejecutar sobre plataformas de cómputo paralela, a continuación presentamos los indicadores que se han utilizado. En este sentido, el

indicador más importante para los algoritmos paralelos es sin ninguna duda el *speedup*, que compara el tiempo de la ejecución secuencial con el tiempo correspondiente para el caso paralelo a la hora de resolver un problema. Si notamos por T_m el tiempo de ejecución para un algoritmo usando m procesadores, el speedup es el ratio entre la ejecución más rápida en un sistema mono-procesador T_1 y el tiempo de ejecución en m procesadores T_m:

$$s_m = \frac{T_1}{T_m} \tag{3.22}$$

Para algoritmos no deterministas no podemos usar esta métrica directamente. Para esa clase de métodos, se debería comparar el tiempo *medio* secuencial con el tiempo *medio* paralelo:

$$s_m = \frac{E[T_1]}{E[T_m]} \ . \tag{3.23}$$

La principal dificultad con esta medida es que los investigadores no se ponen de acuerdo en el significado de T_1 y T_m. En un estudio realizado por Alba [14] se distingue entre diferentes definiciones de speedup dependiendo del significado de esos valores (véase la Tabla 3.1).

Tabla 3.1: Taxonomía de las medidas de speedup propuesta por Alba [14].

- I. Speedup Fuerte
- II. Speedup Débil
  - A. Speedup con parada por calidad de soluciones
    1. Versus panmixia
    2. Ortodoxo
  - B. Speedup con un esfuerzo predefinido

El *speedup fuerte* (tipo I) compara el tiempo de ejecución paralelo respecto al mejor algoritmo secuencial. Esta es la definición más exacta de speedup, pero debido a lo complicado que es conseguir el algoritmo más eficiente actual la mayoría de los diseñadores de algoritmos paralelos no la usan. El *speedup débil* (tipo II) compara el algoritmo paralelo desarrollado por el investigador con su propia versión secuencial. En este caso se puede usar dos criterios de parada: por la calidad de soluciones y por máximo esfuerzo. El autor de esta taxonomía descarta esta última definición debido a que compara algoritmos que no producen soluciones de similar calidad, lo que va en contra del espíritu de esta métrica. Para el speedup débil con parada basada en la calidad de soluciones se proponen dos variantes: comparar el algoritmo paralelo con la versión secuencial canónica (tipo II.A.1) o comparar el tiempo de ejecución del algoritmo paralelo en un procesador con el tiempo que tarda el mismo algoritmo pero en m procesadores (tipo II.A.2). En el primero es claro que se están comparando dos algoritmos diferentes.

Aunque el speedup es la medida más utilizada, también se han definido otras métricas que pueden ser útiles para medir el comportamiento del algoritmo paralelo. Presentamos aquí las dos que hemos utilizado en esta tesis, la eficiencia paralela y la fracción serie.

La *eficiencia* (Ecuación 3.24) es una normalización del speedup que muestra un valor entre 0 y 1 indicando el grado de aprovechamiento de los m procesadores:

$$e_m = \frac{s_m}{m} \ . \tag{3.24}$$

Finalmente, Karp y Flatt [131] desarrollaron una interesante métrica para medir el rendimiento de cualquier algoritmo paralelo que puede ayudarnos a identificar factores más sutiles que los proporcionados por el speedup por sí sólo. Esta métrica se denomina *fracción serie* de un algoritmo (Ecuación 3.25).

$$f_m = \frac{1/s_m - 1/m}{1 - 1/m} \ . \tag{3.25}$$

Idealmente, la fracción serie debería permanecer constante para un algoritmo aunque se cambie la potencia de cálculo de la plataforma donde se ejecuta. Si el speedup es pequeño pero el valor de la fracción serie se mantiene constante para diferentes números de procesadores (m), entonces podemos concluir que la perdida de eficiencia es debida al limitado paralelismo del programa. Por otro lado, un incremento ligero de f_m puede indicar que la granularidad de las tareas es demasiado fina. Se puede dar un tercer escenario en el que se produce una reducción significativa en el valor de f_m, lo que indica que alguna clase de speedup superlineal.

3.5.3. Análisis estadístico de los resultados

Una vez definidos los indicadores de calidad y rendimiento, y realizadas las, al menos, 30 ejecuciones independientes, tenemos un conjunto de datos por cada indicador. Desde el punto de vista estadístico, estos datos se pueden considerar como muestras de una función de densidad de probabilidad y, para poder sacar conclusiones correctas, se ha realizado el siguiente análisis estadístico [69, 225].

Primero aplicamos un test de Kolmogorov-Smirnov para determinar si los valores obtenidos siguen una distribución normal (de Gauss). Si la siguen, utilizamos el test de Levene para comprobar la homocedasticidad de las muestras (igualdad de las varianzas). Si este test es positivo (las varianzas son iguales), se realiza un test ANOVA; en caso contrario, usamos el test de Welch. Para distribuciones no gaussianas, empleamos el test no paramétrico de Kruskal-Wallis para comparar las medianas de los resultados. El nivel de confianza en todos los tests es de 95 % (i.e., un nivel de significancia de 5 % o p-value por debajo de 0,05). La Figura 3.12 muestra gráficamente el procedimiento seguido.

Cuando la comparación de resultados para un problema concreto involucra a más de dos algoritmos, también utilizamos un test de comparaciones múltiples [111]. En concreto, hemos utilizado la función `multcompare` que proporciona Matlab ©, que es capaz de seleccionar el valor crítico más apropiado en la comparación múltiple, atendiendo a las muestras. Los tests que se aplican van desde el método más conservativo *HSD* or *Tukey-Kramer* hasta el menos conservativo *Scheffe's S*. Se ha mantenido el mismo nivel de confianza que el usado en la fase anterior ($\alpha = 0,05$).

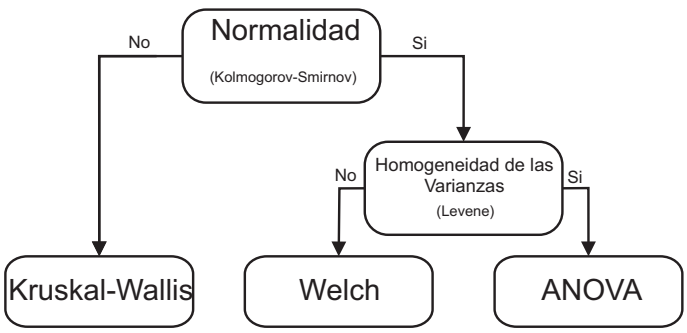


Figura 3.12: Análisis estadístico de los experimentos.

3.6. Conclusiones

En este capítulo hemos ofrecido una introducción al campo de las metaheurísticas. En primer lugar, hemos dado las definiciones previas y hemos introducido una extensión de la definición formal de metaheurística. En segundo lugar, hemos incluido un repaso por las técnicas más importantes y populares dentro de este campo. Estas breves descripciones han sido realizadas siguiendo una clasificación de las metaheurísticas que las dividen en dos clases, atendiendo al número de soluciones tentativas con la que trabajan en cada iteración: metaheurísticas basadas en trayectoria y en población.

Debido a las características de los problemas del mundo real que se pretender resolver en esta tesis, también hemos introducido los conceptos básicos de optimización multiobjetivo, ya que varios de estos problemas tienen varias funciones que han de ser optimizadas a la vez. Después, hemos introducido las metaheurísticas paralelas indicando las distintas formas de paralelismo consideradas en la literatura para cada clase de metaheurística (trayectoria y población). Esto ha estado motivado por las grandes necesidades de cómputo que se necesita para abordar los problemas considerados. Hemos presentado después cómo las metaheurísticas pueden abordar problemas de optimización computacionalmente muy costosos utilizando sistemas de computación grid.

Para terminar, hemos detallado el procedimiento estadístico utilizado para comparar de forma estadísticamente correcta los resultados obtenidos.

Parte II

Modelado de problemas

Capítulo 4

Planificación de celdas en redes de telefonía móvil

El primer problema que hemos abordado en esta tesis es el de la planificación de celdas en redes de telefonía móvil o ACP (*Automatic Cell Planning*), más concretamente, en el subsistema de radio de este tipo de sistemas de telecomunicaciones [178]. El problema consiste en localizar sitios para posicionar estaciones base de la red así como definir los parámetros para estas estaciones base, de forma que se satisfagan ciertos requisitos propios de la red, como el máximo cubrimiento de área y capacidad de tráfico, a la vez que se minimiza el coste de la infraestructura. La configuración de estas estaciones base no es una tarea trivial ya que implica seleccionar el tipo de antena que se va a utilizar (varias antenas direccionales o bien una única antena omnidireccional), además de la potencia de emisión y los ángulos de inclinación y acimut.

Llegar a un buen diseño para la red de radio supone tener en cuenta varios factores que compiten entre sí. Por ejemplo, el coste se puede reducir teniendo unas pocas antenas omnidireccionales funcionando a máxima potencia. Así, se alcanzaría una buena cobertura con escaso solapamiento entre las celdas y, por tanto, con pocas interferencias. Sin embargo, la red seguramente no podrá satisfacer la demanda de tráfico de cada celda. Para tratar de solventar este problema se requieren más antenas, lo que conlleva un incremento en el coste de la red así como interferencias potencialmente mayores. Hay que buscar, por tanto, soluciones de compromiso entre estos objetivos contrapuestos. En este sentido, además de utilizar un modelo del problema muy cercano a la realidad (con una base de datos geográfica de localizaciones y simulación de la propagación de las ondas electromagnéticas), el problema se ha formulado como un problema multiobjetivo con restricciones en el que hay que minimizar el coste (número de antenas de la red) y maximizar la calidad del servicio (maximizar el tráfico de la red y minimizar las interferencias), cumpliendo siempre con unas restricciones mínimas de cobertura y continuidad del servicio cuando el usuario cambia de celda (*handover*). El modelado de este problema, que está basado en las tecnologías de segunda generación GSM 900 y DSC 1800 así como en UMTS (tecnología de tercera generación), se complementa con datos de redes de telefonía móvil reales proporcionadas por empresas del sector.

La estructura del capítulo es como sigue. En primer lugar, detallamos el esquema de las redes GSM para introducir la terminología propia del campo así como los componentes principales del sistema. A continuación, en la Sección 4.2, se detalla la formulación del problema que vamos a utilizar en esta tesis, incluyendo todos los modelos usados para el área de trabajo, la propagación de las señales, etc. Esta sección incluye, además, un análisis de la complejidad del problema y del tamaño del espacio de búsqueda. El capítulo termina con las principales conclusiones alcanzadas en las Sección 4.3.

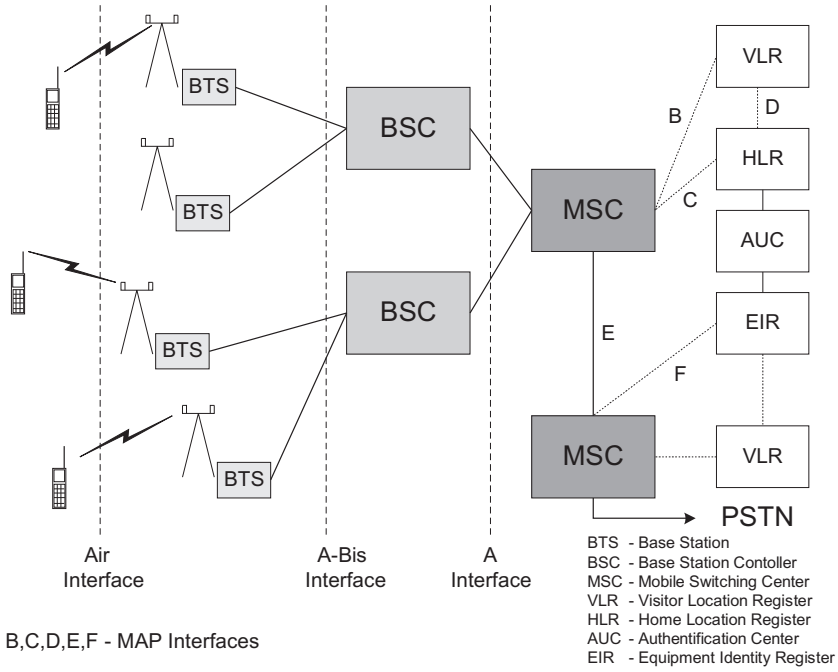


Figura 4.1: Esquema de la arquitectura GSM.

4.1. El sistema GSM

El esquema de la arquitectura del sistema GSM se muestra en la Figura 4.1. Como se puede observar, una red GSM está compuesta por muchos componentes diferentes, siendo aquellos que tienen que ver con la red de radio son las estaciones base o *Base Transceiver Station* (BTS) y los transmisores/receptores o *transceivers* (TRXs). En esencia, una BTS se puede considerar como un conjunto de TRXs. La principal función de un TRX 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 terminal móvil y la red GSM. Las BTSs se organizan normalmente en sectores, siendo típica la instalación de uno a tres sectores por BTS. Cada sector define lo que se denomina celda (zona de geográfica donde la señal emitida por el sector es la da servicio a los terminales móviles).

Las líneas continuas que conectan componentes en la Figura 4.1 llevan tanto tráfico (voz y datos) como señalización de control. Las líneas punteadas son sólo líneas de señalización. La información que se intercambia en esas conexiones es necesaria para soportar movilidad de usuarios, características de red, funcionamiento y mantenimiento, autenticación, cifrado de datos y muchas otras funciones necesarias para el correcto funcionamiento de la red.

4.2. Modelado y formulación del problema

El problema ACP se ha modelado utilizando la estrategia más avanzada presentada en la Sección 2.1, que está basada en la discretización del terreno en varios conjuntos de puntos de test (de recepción, de servicio y de tráfico). También incluye un conjunto de emplazamientos candidatos del que hay que seleccionar aquéllos en los que se colocarán las BTSs (no está permitido ubicarlas en cualquier sitio, como ocurren en los casos reales) [216, 217]. A continuación se especifican los modelos utilizados, así como los objetivos y las restricciones del problema.

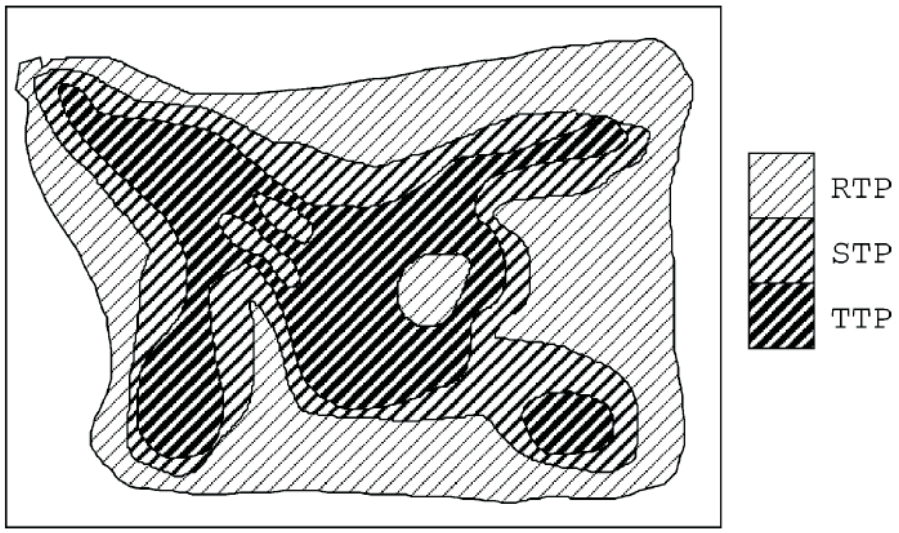


Figura 4.2: Relación en el área de trabajo de los puntos de test de recepción (RTP), de servicio (STP) y de tráfico (TTP).

4.2.1. Área de trabajo y elementos de la red

En esta sección se introducen el modelo para el área geográfica y los componentes del sistema de telefonía que se utilizan en la definición del problema. Así, el área de trabajo $\mathcal{P}$ es la zona geográfica donde se va a desplegar la red y se describe con la ayuda de una base de datos de mapas digitales. Este área se discretiza y se definen en ella cuatro conjuntos de puntos de test:

- Un conjunto de emplazamientos candidatos para el posicionamiento de las BTSs, $\mathcal{L} = \{L_i/i \in [1, \ldots, m]\}$, donde m representa el número de emplazamientos, que quedan definidos por sus coordenadas geográficas (x, y) y su altura sobre el nivel del mar z, $L_i = (x_i, y_i, z_i)$.

- Un conjunto de puntos de test de recepción o RTPs (*Reception Test Points*) donde se comprueba la señal de radio, $\mathcal{R} = \{R_i/i \in [1, \ldots, l]\}$, $R_i \in \mathbb{R}^3$, siendo l es el número de RTPs. Cada RTP se puede utilizar para calcular el cubrimiento de la red.

- Un conjunto de puntos de test de servicio o STPs (*Service Test Points*) en los que se evalúa el servicio, $\mathcal{ST} = \{ST_i/i \in [1, \ldots, k]\}$, donde k representa el número máximo de STPs. $\mathcal{ST}$ define el conjunto de puntos donde la red debe superar un umbral mínimo de calidad para asegurar la calidad del servicio. Este umbral, S_{q_i}, depende de los terminales móviles.

- Un conjunto de puntos de test de tráfico o TTPs (*Traffic Test Points*) donde se comprueba la cantidad de tráfico, $\mathcal{T} = \{TT_i/i \in [1, \ldots, n]\}$, siendo n el número de TTPs. Cada uno de estos TTPs se asocia con la cantidad de tráfico previsto en este punto, e_i, en Erlangs (unidad para medir el tráfico). Dependiendo de la cantidad total de tráfico en sus TTPs, cada BTS tendrá instalados un número diferente de TRXs.

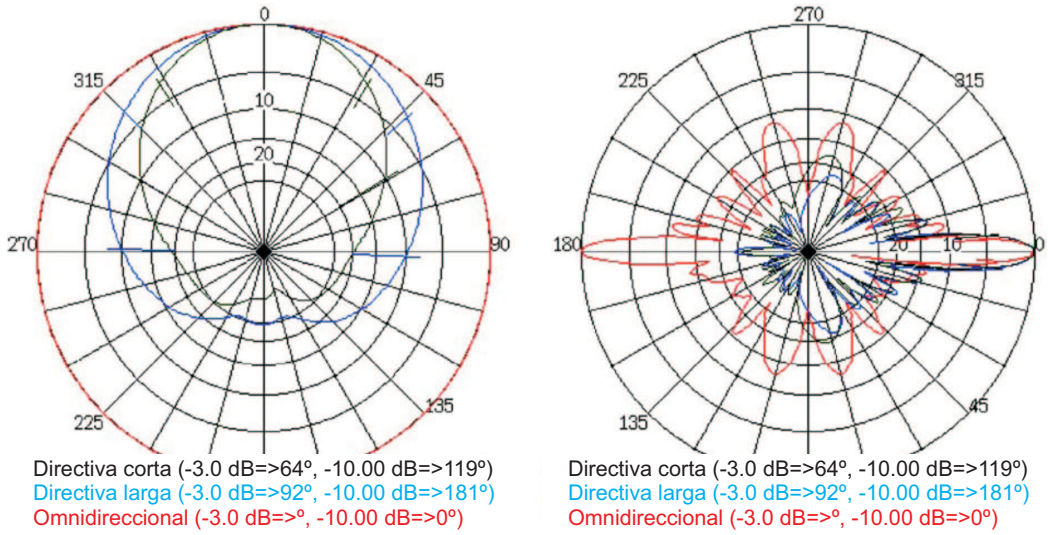


Figura 4.3: Diagramas de radiación de los tres tipos de antena.

Para asegurar una aceptable calidad de la señal en el área donde se localiza el tráfico, cada punto de test de tráfico, TT_i, es a su vez un punto de test de servicio, ST_i. Además, es necesario saber la señal de radio en cada ST_i, por lo que también se asocia cada ST_i a un R_i. De esta forma, la siguiente condición siempre se cumple:

$$\mathcal{T} \subseteq \mathcal{ST} \subseteq \mathcal{R} \ , \tag{4.1}$$

lo que es una caracterización adicional de los tres conjuntos de puntos de test. La Figura 4.2 muestra un ejemplo de un área de trabajo real y la inclusión entre STPs y TTPs.

Teniendo en cuenta este modelo del área geográfica, ahora se presentan los modelos de los sistemas que intervienen en la red de telefonía y que van a completar la definición del problema.

- BTSs. En cada emplazamiento L_i, $L_i \in \mathcal{L}$, se pueden establecer una o varias BTSs dependiendo del tipo se instale. Se pueden utilizar tres tipos diferentes: omnidireccionales, de directiva corta y de directiva larga, que difieren en su diagramas de radiación (Figura 4.3). Cada BTS puede tener instalados varios TRXs para cubrir las demandas de tráfico, hasta un máximo de 7, que se corresponde con 43 Erlangs [179]. En general, un emplazamiento tiene una única BTS omnidireccional o de una a tres BTSs directivas para dar cobertura a un área determinada.

- Celdas. Una celda es el conjunto de STPs que tienen a una BTS dada como mejor servidora de señal (la potencia de la señal recibida de esta BTS es la mayor). Formalmente, la celda C_{jk} asociada a la BTS k-ésima del emplazamiento L_j se puede definir como:

$$C_{jk} = \{ST_i | Cd_{ijk} \geq Sq_i \text{ y } Cd_{ijk} \geq Cd_{iuw}, \forall u \in \mathcal{L}, \forall w \in \mathcal{B}\} \tag{4.2}$$

  donde Cd_{ijk} denota la potencia de la señal medida en ST_i que transmite la BTS k-ésima del emplazamiento L_j, Sq_i representa el umbral de servicio de ST_i y $\mathcal{B}$ es el conjunto de BTSs instaladas en toda la red. En la práctica, la forma de las celdas no es homogénea ya que depende tanto de la topología de la zona geográfica, como del modelo de propagación. La

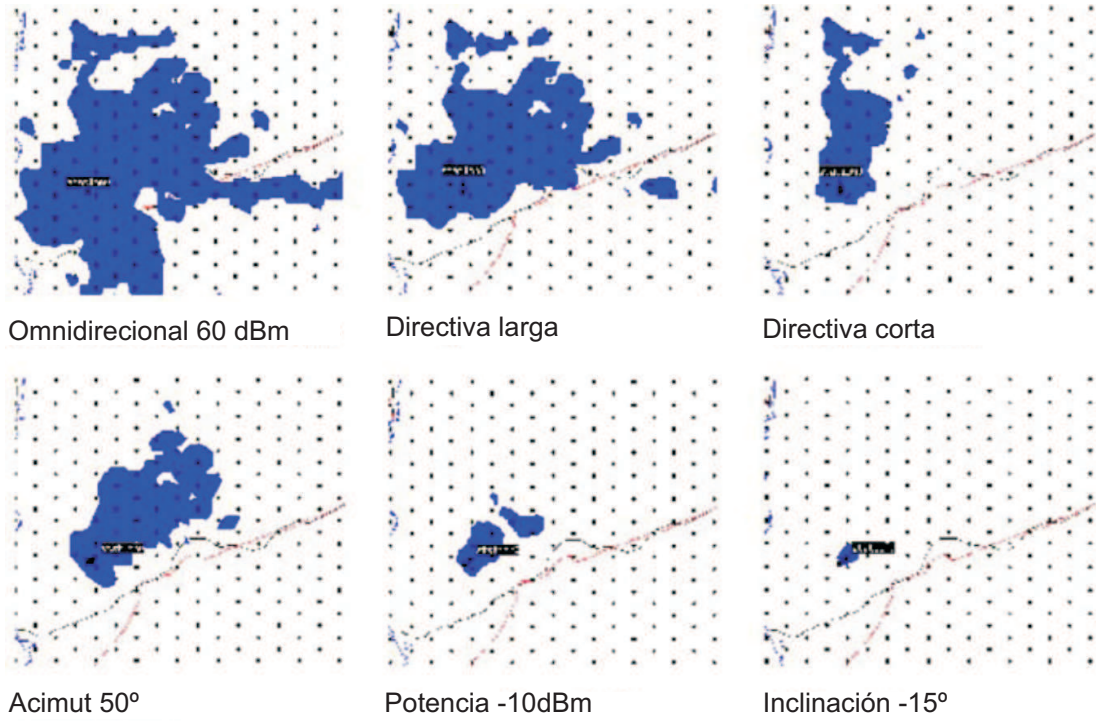


Figura 4.4: Forma de la celda de una BTS cuando se modifican sucesivamente su tipo (de omnidireccional a directiva larga y a directiva corta), el acimut (50°), la potencia de emisión (-10 dBm) y el ángulo de inclinación (-15°).

Figura 4.4 muestra la forma de una celda para diferentes configuraciones de la antena en la que se va cambiando sucesivamente el tipo de antena, el acimut, la potencia de emisión y el ángulo de inclinación.

- Modelo de propagación. Los obstáculos y superficies reflectantes que se encuentran en las proximidades de una antena tienen una influencia sustancial en la propagación de las ondas electromagnéticas [155]. Además, las características de propagación varían de un lugar a otro y, debido a la movilidad de los dispositivos, también en el tiempo. Por tanto, la trasmisión entre transmisor y receptor puede variar desde una línea directa de visión, hasta verse muy obstaculizada por edificios, vegetación e incluso el propio terreno. Existen varios modelos de propagación que, junto con bases de datos de mapas digitales, se pueden utilizar para calcular la potencia de la señal en una zona geográfica concreta, como el de espacio libre (*free-space*), Hokumura-Hata o Walfish-Ikegami [50]. Estos modelos se diferencian en la información que utilizan para calcular la fuerza de campo electromagnético (*field strength*). Cuanto más preciso es un modelo, más costoso es en términos computacionales [123]. El modelo utilizado aquí es el de propagación en espacio libre, que se puede considerar como esqueleto para otros más complejos.

4.2.2. Objetivos y restricciones

Dada una posible solución de diseño de una red candidata, se consideran tres objetivos que han de ser optimizados: minimizar el coste de la red (número de emplazamientos), maximizar la capacidad (tráfico), y minimizar las interferencias (o maximizar la calidad de las señales de radio). Estos términos se describen a continuación:

- Coste. La minimización del coste de la red se consigue ocupando el menor número de emplazamientos posibles, independientemente del número de BTSs que se instalen en cada emplazamiento. Por tanto, se define f_{coste} como:

$$f_{coste} = \sum_{j \in \{1,\ldots,m\}} y_j, \tag{4.3}$$

  donde

$$y_j = \begin{cases} 1 & \text{si se coloca al menos una BTS en } L_j \ , \\ 0 & \text{en otro caso} \ . \end{cases}$$

- Tráfico. Para que una red sea eficiente, el tráfico que puede soportar debe ser maximizado. Dado que los TTPs nos dan una estimación de la cantidad de tráfico de cada celda y que, según el equipamiento utilizado, no se pueden instalar más de 7 TRXs por antena con una capacidad máxima de 43 Erlangs, la función objetivo f_{traf} (a maximizar) se puede expresar como:

$$f_{traf} = \sum_{j \in \{1,\ldots,m\}} \sum_{k=1}^{3} \mathcal{H}_{jk}, \tag{4.4}$$

  donde $\mathcal{H}_{jk}$ representa el tráfico soportado por la k-ésima BTS del emplazamiento L_j, B_{jk}:

$$\mathcal{H}_{jk} = \text{máx} \left\{ 43, \sum_{i \in \{1,\ldots,n\}} e_i x'_{ijk} \right\} \tag{4.5}$$

  siendo e_i el tráfico en el punto TT_i y

$$x'_{ijk} = \begin{cases} 1 & \text{si el punto de test } ST_i \text{ está cubierto por } B_{jk} \ , \\ 0 & \text{en otro caso} \ . \end{cases}$$

- Interferencias. El solapamiento entre diferentes celdas provoca la aparición de interferencias que hacen bajar la calidad del servicio que la red proporciona. Así, la interferencia total es la suma de las interferencias calculadas en cada STP del área de trabajo $\mathcal{P}$. En cada punto, las BTSs de la red que generan los h[1] campos más potentes se consideran como servidores potenciales y, por tanto, útiles. Por esta razón son descartados en el cálculo de las interferencias. Cada STP no necesita más de h BTSs consideradas como potenciales servidores: el mejor servidor y $h-1$ BTSs para realizar *handover*. El resto de señales cuya campo está por debajo de los h primeros se consideran como inútiles y problemáticos (Figura 4.5). El tercer objetivo consiste en minimizar la suma de las interferencias asociadas a BTSs para cada STP, pero que no son BTSs para *handover* y cuya señal está por encima del umbral de sensibilidad de los teléfonos móviles. Así, por cada punto de servicio ST_i, los campos de intensidad de señal Cd se pueden ordenar de acuerdo con su fuerza:

$$\begin{aligned} & Cd_{iu_1v_1} \geq Cd_{iu_2v_2} \geq \cdots \geq Cd_{iu_hv_h} \\ & \geq \underbrace{Cd_{iu_hv_h} \geq Cd_{iu_{h+1}v_{h+1}} \geq \cdots}_{I_{h_i}} \geq S_m \ , \end{aligned}$$

[1] Este parámetro ha sido fijado por ingenieros expertos en el campo y tiene un valor de 4

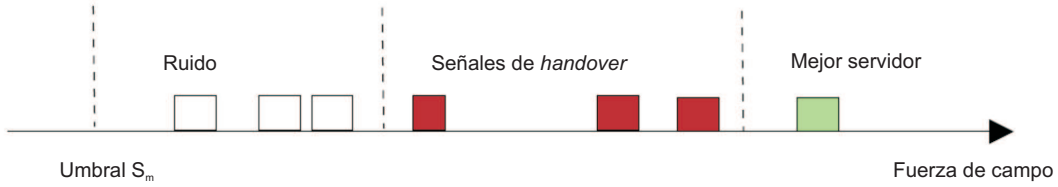


Figura 4.5: Las señales se consideran de forma diferente en cada punto de recepción: mejor señal, señales de *handover* e interferencias.

donde S_m denota el umbral de sensibilidad del receptor y h define el número de campos útiles de servidores *handover*. Asimismo, Cd_{ijk} se corresponde con la máxima señal recibida en el punto ST_i desde la k-ésima BTS del emplazamiento L_j y, por tanto, $Cd_{iu_hv_h}$ tal que $h > 4$ forma el conjunto I_{h_i} de interferentes. El tercer objetivo, f_{interf}, se puede formular entonces como:

$$f_{interf} = \sum_{i\in\{1,\ldots,k\}} \sum_{Cd_{iuv}\in I_{h_i}} (Cd_{iuv} - S_m) \quad . \tag{4.6}$$

Además de estos objetivos, toda red debe satisfacer dos restricciones relativas a la cobertura y el *handover*. En el primer caso, la configuración resultante debe garantizar que existe un área geográfica mínima a la que se le debe dar servicio con calidad y, el segundo, trata de garantizar que se puedan transferir llamadas de una celda a otra con garantías de continuidad.

- Cobertura. Para garantizar el cubrimiento del área geográfica, todos los STPs deben recibir, al menos, una señal de radio de intensidad superior al umbral de sensibilidad de los terminales móviles. Por tanto, la unión de todas las celdas C_{jk} debe ser igual al conjunto de STPs:

$$\bigcup_{j,k} C_{jk} = \mathcal{ST}. \tag{4.7}$$

En términos numéricos, el porcentaje de área cubierta se puede expresar como

$$cob = \frac{1}{|\mathcal{ST}|} \sum_{i\in\{1,\ldots,|\mathcal{T}|\}} \max_{\substack{j\,\in\,\{1,\ldots,|\mathcal{L}|\}\\ k\,\in\,\{1,2,3\}}} \{x_{ijk}\}\,, \tag{4.8}$$

donde

$$x_{ijk} = \begin{cases} 1 & \text{si el punto de servicio } ST_i \text{ está cubierto por } B_{jk}, \\ & Cd_{ijk} = \max\limits_{1\le h\le 3}\left\{\max\limits_{1\le l\le|\mathcal{L}|} Cd_{ilh}\right\}, \\ & Cd_{ijk} \ge Sq_i, \text{ y } Cu_{ijk} \ge Sq_i, \\ 0 & \text{en otro caso} \ , \end{cases}$$

donde Cu_{ijk} es la fuerza de la señal de subida (del terminal móvil a la antena). La Ecuación 4.8 calcula, por tanto, el porcentaje del área de trabajo cubierta, i.e, $cob \in [0, 1]$. La restricción c_1 se define entonces de la siguiente forma:

$$c_1 = \begin{cases} 1 - cob & \text{si } cob \le 0{,}8 \\ 0 & \text{en otro caso} \ , \end{cases}$$

es decir, c_1 se satisface con una cobertura superior al 80 % y la violación es máxima cuando la cobertura es próxima a cero.

- Handover. Una red de telefonía móvil debe asegurar la continuidad en las comunicaciones desde una celda inicial hasta una celda destino cuando el usuario se mueve (*cell switching*). El mecanismo que garantiza esta continuidad es el *handover*. Cuando un terminal móvil pasa de una celda a otra, la celda inicial corta la comunicación tan pronto como la celda destino es capaz de garantizar continuidad. Este mecanismo requiere que todas las celdas dispongan de un área de handover no vacía (solapamiento) donde la diferencia entre las dos señales recibidas más potentes sea inferior a un cierto umbral (fijado aquí a 7 dBm). Sea HOC_{jk} el área de solapamiento de la celda C_{jk}, que se define como

$$\begin{aligned} HOC_{jk} = \quad & \{ST_i \in C_{jk} | \exists (j,k)\ Cd_{ijk} \geq Sq_i, \exists (j',k')\ Cd_{ij'k'} \geq Sq_i, \\ & (j,k) \neq (j',k'),\ \text{tal que}\ |Cd_{ijk} - Cd_{ij'k'}| \leq 7\}\ . \end{aligned} \tag{4.9}$$

  Esta restricción mide que $HOC_{jk} \neq \emptyset$. Así, sea HC_i una variable que es igual a 1 si la BTS i tiene un área de solapamiento no vacía y 0 en otro caso. Numéricamente, este área de solapamiento se puede expresar como:

$$ho = \sum_{i \in \mathcal{B}} \frac{HC_i}{|\mathcal{B}|} \tag{4.10}$$

  donde $\mathcal{B}$ es el conjunto de BTSs de todas las BTSs de la red. La segunda restricción, c_2, quedaría, por tanto, de la siguiente forma:

$$c_2 = \begin{cases} 1 - ho & \text{si } ho \leq 0{,}5 \\ 0 & \text{en otro caso} \end{cases},$$

  es decir, se satisface cuando hay solapamiento entre más del 50 % de las celdas, y la violación es máxima cuando este solapamiento tiende a cero.

4.2.3. Espacio de búsqueda y complejidad

El problema de la planificación de celdas es NP-duro, lo que se ha demostrado mediante reducciones polinomiales en [245]. La principal variable de decisión del problema consiste en colocar las BTSs en los emplazamientos disponibles. Como ya se ha mencionado, cada BTS puede ser omnidireccional, de directiva corta y de directiva larga, de forma que en cada emplazamiento se puede instalar una única BTS omnidireccional o de una a tres BTSs direccionales.

Además de seleccionar apropiadamente los emplazamientos, hay que configurar tres parámetros de cada BTS: potencia, inclinación y acimut, que, si bien son valores reales, se discretizan como aparece en la Tabla 4.1. Así, una antena omnidireccional tiene 15 configuraciones posibles (sólo el parámetro de potencia, ya que no necesita los ángulos de inclinación y de acimut), mientras que para cada antena direccional se abren 3.240 posibilidades (potencia × inclinación × acimut). Entonces, para una BTS cualquiera, tenemos 11.358.415.170 configuraciones potenciales. Si la instancia en cuestión tiene N emplazamientos candidatos, hay que encontrar una combinación de BTSs adecuada de entre $2^{N \times 33}$, un espacio de búsqueda enorme aún para redes pequeñas.

Por último, queremos destacar que el cálculo de los objetivos y restricciones anteriores conlleva tareas pesadas de cómputo. El coste computacional depende principalmente del número de STPs y del número de BTSs de la red ya que, por cada STP, hay que predecir la fuerza de campo desde cada BTS de la red. El cálculo de esta predicción requiere, además, el uso de funciones transcendentes computacionalmente costosas como el arco tangente.

Parámetro	Dominio	Discretización	Valores distintos
Potencia	[26 dBm, 55 dBm]	2 dBm	15
Inclinación	[-15º, 0º]	3º	6
Acimut	[0º,360º]	10º	36

Tabla 4.1: Discretización de los parámetros de las BTSs.

4.3. Conclusiones

Este capítulo está dedicado a la formulación detallada del problema de la planificación de celdas o ACP. Hemos modelado los componentes necesarios del sistema de telefonía móvil que es relevante para el problema, así como el área de trabajo. Todos los modelos usados en la formulación se acercan a la realidad, de forma que los algoritmos se enfrentarán a problemas reales y no a abstracciones simples del problema. Hemos analizado también brevemente la complejidad del problema y hemos mostrado el enorme espacio de búsqueda asociado, aún cuando se resuelvan instancias pequeñas.

Capítulo 5

Asignación automática de frecuencias en redes GSM

El éxito del sistema de comunicaciones GSM (*Global System for Mobile communication*) [186] radica en usar eficientemente el escaso espectro de radio disponible (la estructura del sistema GSM se describió en la Sección 4.1). GSM utiliza los esquemas FDMA (*Frequency Division Multiple Access*) y TDMA (*Time Division Multiple Access*) para mantener varios enlaces de comunicación en paralelo. Así, la banda de frecuencia disponible se particiona en canales (o frecuencias) que deben ser asignadas a los *transceivers* (TRXs) instalados en las estaciones base de la red. Para el correcto funcionamiento de este tipo de redes con el mínimo coste posible, es necesario optimizar el uso de los canales disponibles. A este problema se le conoce como la Planificación Automática de Frecuencias (AFP), el Problema de la Asignación de Frecuencias (FAP) o el Problema de la Asignación de Canales (CAP). Bajo estos términos generales se pueden encontrar diferentes versiones del problema para los que se han propuesto bastantes modelos matemáticos desde finales de los años 60 [1, 82, 138]. No obstante, en este capítulo nos centramos en modelos y conceptos que se aplican a la planificación de frecuencias en GSM. En este tipo de redes, los operadores disponen normalmente de un pequeño número de frecuencias (algunas docenas) para satisfacer la demanda de miles de TRXs, haciendo la reutilización de frecuencias, por tanto, inevitable. Sin embargo, esta reutilización está limitada por las interferencias que ellas mismas provocan, haciendo que la calidad del servicio para los usuarios se reduzca hasta niveles no satisfactorios. Consecuentemente, la planificación de frecuencias es una tarea de gran importancia para los operadores GSM, y no sólo en el despliegue inicial del sistema, sino también en expansiones/modificaciones posteriores, a la hora de resolver informes de interferencias no previstos, y/o adelantarse a situaciones conocidas (e.g., un incremento de la demanda de tráfico en algunas zonas de la red, como por ejemplo las regiones costeras en verano, etc.).

Como generalización del problema de coloreado de grafos, el problema AFP es NP-duro [105] ya en sus modelos matemáticos más simples. Como se ha mencionado anteriormente, hemos trabajado con el problema de la asignación de frecuencias dentro del marco de las redes GSM [79, 158]. Para el trabajo realizado en esta tesis hemos desarrollado una nueva formulación matemática del problema que utiliza, tal y como se hace en la industria, información muy precisa de las interferencias existentes en redes GSM reales [159, 161]. El planteamiento dista bastante de los problemas de asignación de frecuencia clásicos existentes en la literatura, que están considerados en su mayoría como bancos de pruebas. Esto nos permite enfrentar nuestras propuestas algorítmicas a un problema del mundo real.

El resto del capítulo está organizado como se indica a continuación. La Sección 5.1 incluye algunos conceptos propios de la planificación de frecuencias en redes GSM. Nuestra propuesta de formulación matemática se detalla en la Sección 5.2. Las conclusiones de este capítulo aparecen en la Sección 5.3.

5.1. Planificación de frecuencias en redes GSM

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 otras cuestiones: dónde instalar las BTSs o cómo configurar los parámetros de propagación de las antenas (inclinación, acimut, potencia, etc.), entre otros [179]. Una vez que tanto la localización de las BTSs como la configuración de cada sector están decididos (problema ACP, ver Capítulo 4), hay que determinar el número de TRXs que se instalará en cada sector. Este número depende de la demanda de tráfico que el sector correspondiente ha de soportar. La planificación de frecuencias consiste en asignar un canal (frecuencia) a cada TRX [79]. El problema de optimización surge ya que el espectro de radio que se puede usar es escaso y, consecuentemente, los TRXs de la red han de reutilizar frecuencias.

Sin embargo, esta reutilización de frecuencias puede causar interferencias que reducen la calidad del servicio proporcionada por la red. Así, si se usa la misma frecuencia o frecuencias adyacentes en celdas vecinas que están solapadas, las interferencias que se provocan son muy significativas. La cuestión aquí es que calcular este nivel de interferencia es una tarea difícil que no sólo depende de los canales utilizados, sino también de las señales de radio y de las propiedades del entorno. Está claro que, a medida que la información sobre las interferencias de una red GSM es más precisa, mejor será el plan de frecuencias que se puede calcular para dicha red. Existen diferentes formas de cuantificar las interferencias que van desde métodos teóricos hasta mediciones extensivas [142]. El resultado de todas estas técnicas es lo que se denomina matriz de interferencias (*interference matrix*), M. Cada elemento $M(i, j)$ de M indica la degradación de la calidad de la señal si las celdas i y j tienen asignada la misma frecuencia. Esto se conoce como interferencia co-canal (*Co-channel interference*). Además de este tipo de interferencia, también pueden producirse interferencias por canal adyacente (*adjacent-channel interference*) que están provocadas por TRXs que operan con canales adyacentes, es decir, un TRX tiene asignada la frecuencia f y el otro TRX tiene la frecuencia $f + 1$ o $f - 1$ (Figura 5.1). Tener, por tanto, una matriz de interferencia muy precisa es esencial para la planificación de frecuencias en redes GSM, ya que el algoritmo de asignación tiene como objetivo final minimizar las interferencias de la red basándose en la información contenida en dicha matriz.

5.2. Un nuevo modelo matemático para el problema AFP

En la literatura se pueden encontrar multitud de formulaciones matemáticas para modelar problemas tipo AFP [1] y, más concretamente, problemas de asignación de frecuencias en redes GSM [30, 79]. Algunos de los modelos existentes incluyen incluso aspectos específicos de GSM como los saltos de frecuencia (frequency hopping) [27, 185]. Sin embargo, ninguna de estas formulaciones considera conceptos y otras tecnologías avanzadas que se usan en la planificación de frecuencias real en redes GSM [80]. En general, para tratar con el problema real, los investigadores han de usar simuladores, como ocurre en [62, 125, 222].

La nueva característica de nuestra propuesta reside en la información contenida en la matriz de interferencias. El modelo tiene como objetivo medir el rendimiento global de las señales usadas en las comunicaciones dentro de la red GSM. Hasta ahora, las matrices de interferencia incluían sólo un único valor para caracterizar las interferencias entre pares de TRXs (normalizado típicamente entre

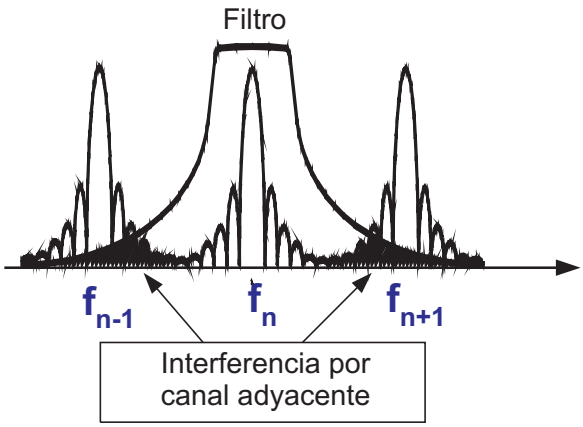


Figura 5.1: Interferencias por canal adyacente.

cero y uno). Por el contrario, la matriz de interferencia que se utiliza en este nuevo modelo muestra las interferencias entre pares de celdas, pero dando en su lugar la distribución de probabilidad completa del ratio portadora/interferente o ratio C/I (*carrier/interferer ratio*). Esta definición está importada directamente de la planificación de frecuencias en redes GSM reales tal y como se realiza en la industria (y no está generada por ordenador mediante el muestreo de variables aleatorias), ya que no sólo permite calcular planes de frecuencia muy precisos sino que también se utiliza para predecir la calidad del servicio que proporcionará la red.

La formulación que se propone es como sigue. Sea $T = \{t_1, t_2, \ldots, t_n\}$ un conjunto de n TRXs, y sea $F_i = \{f_{i1}, \ldots, f_{ik}\} \subset \mathbb{N}$ el conjunto de frecuencias válidas que se pueden asignar al TRX $t_i \in T$, $i = 1, \ldots, n$. Es importante indicar que k, la cardinalidad de F_i, no es necesariamente la misma para todos los TRXs. Además, sea $S = \{s_1, s_2, \ldots, s_m\}$ un conjunto de sectores (o celdas) de cardinalidad m. Cada TRX $t_i \in T$ está instalado en exactamente uno de los m sectores y, de ahora en adelante, denotaremos con $s(t_i) \in S$ al sector en el que el TRX t_i está instalado. Finalmente tenemos la matriz de interferencias, $M = \{(\mu_{ij}, \sigma_{ij})\}_{m \times m}$. Los dos elementos μ_{ij} y σ_{ij} correspondientes a la entrada en la matriz $M\,(i, j) = (\mu_{ij}, \sigma_{ij})$ son valores numéricos mayores o iguales que cero de forma que μ_{ij} representa la media y σ_{ij} la desviación típica de una distribución normal de probabilidad que describe la ratio C/I cuando los sectores i y j operan con la misma frecuencia [249]. Cuanto mayor es el valor de μ_{ij}, menor es la interferencia y, por tanto, mejor la calidad en la comunicación. Hay que indicar que la matriz de interferencia está definida a nivel de sector (celda), ya que todos los TRXs que están instalados en un sector sirven el mismo área.

Una solución al problema se obtiene asignando a cada TRX $t_i \in T$ una de las frecuencias de F_i. Por tanto, una solución (o plan de frecuencias) se denota por $p \in F_1 \times F_2 \times \cdots \times F_n$, donde $p(t_i) \in F_i$ es la frecuencia asignada al TRX t_i. El objetivo es encontrar una solución p que minimice la siguiente función de coste:

$$C(p) = \sum_{t \in T} \sum_{u \in T, u \neq t} C_{\text{sig}}(p, t, u) \ . \tag{5.1}$$

Para definir la función $C_{\text{sig}}(p, t, u)$, sean s_t y s_u los sectores donde están instalados los TRXs t y u, esto es, $s_t = s(t)$ y $s_u = s(u)$, respectivamente. Además, sean $\mu_{s_t s_u}$ y $\sigma_{s_t s_u}$ los elementos correspondientes a la entrada de la matriz $M(s_t, s_u)$, correspondiente a los sectores s_t y s_u.

Entonces:

$$C_{\text{sig}}(p,t,u) = \begin{cases} K & \text{si } s_t = s_u, |p(t)-p(u)| < 2 \\ C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{si } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t)-p(u)| = 0 \\ C_{\text{adj}}(\mu_{s_t s_u}, \sigma_{s_t s_u}) & \text{si } s_t \neq s_u, \mu_{s_t s_u} > 0, |p(t)-p(u)| = 1 \\ 0 & \text{en otro caso.} \end{cases} \tag{5.2}$$

La constance K tiene un valor muy elevado ($K >> 0$) y está definida por el diseñador de la red, de forma que se penalice la asignación de la misma frecuencia o frecuencias adyacentes a TRXs que están sirviendo la misma zona. La función $C_{\text{co}}(\mu,\sigma)$ se define como:

$$C_{\text{co}}(\mu,\sigma) = 100\left(1{,}0 - Q\left(\frac{c_{\text{SH}}-\mu}{\sigma}\right)\right) \ , \tag{5.3}$$

donde

$$Q(z) = \int_z^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx \tag{5.4}$$

es la integral de cola de una distribución de probabilidad Gaussiana de media cero y varianza uno, y c_{SH} es el umbral mínimo de calidad en la señal. La función Q es muy usada en las comunicaciones digitales porque caracteriza la probabilidad de error de señales digitales [226]. Esto significa que $Q\left(\frac{c_{\text{SH}}-\mu}{\sigma}\right)$ es la probabilidad de que el ratio C/I sea mayor que c_{SH} y, por tanto, $C_{\text{co}}(\mu_{s_t s_u}, \sigma_{s_t s_u})$ calcula la probabilidad de que el ratio C/I en el área de servicio del sector s_t esté por debajo del umbral mínimo de calidad debido a las interferencias provocadas por el sector s_u. Así, si esta probabilidad es pequeña, el valor del ratio C/I en el sector s_t no se vería degradada por la señal interferente procedente de s_u y, de esta manera, la calidad de las comunicaciones en esa zona de la red es alta. Por el contrario, una probabilidad elevada —y consecuentemente un elevado coste— hace que el ratio C/I esté principalmente por debajo del umbral mínimo c_{SH}, provocando que la comunicación sea de mala calidad.

Ya que no existe una fórmula cerrada para calcular la integral de la función Q, hay que evaluarla con un método numérico. En este caso hemos utilizado la función de error complementario, E,

$$Q(z) = \frac{1}{2} E\left(\frac{z}{\sqrt{2}}\right) \tag{5.5}$$

para la que existe un método numérico que permite calcular el valor de E con un error inferior a $1{,}2 \cdot 10^{-7}$ [203], que es un error aceptable para nuestro problema. Análogamente, la función $C_{\text{adj}}(\mu,\sigma)$ se define como

$$\begin{aligned} C_{\text{adj}}(\mu,\sigma) &= 100\left(1{,}0 - Q\left(\tfrac{c_{\text{SH}}-c_{\text{ACR}}-\mu}{\sigma}\right)\right) \\ &= 100\left(1{,}0 - \tfrac{1}{2}E\left(\tfrac{c_{\text{SH}}-c_{\text{ACR}}-\mu}{\sigma\sqrt{2}}\right)\right). \end{aligned} \tag{5.6}$$

La única diferencia entre las funciones C_{co} y C_{adj} está en la constante $c_{\text{ACR}} > 0$ (*adjacent channel rejection*) que aparece en la definición de la función C_{adj}. Esta es una constante hardware que mide la capacidad del receptor para recibir una señal en presencia de señales no deseadas en un canal adyacente. Hay que indicar que la constante c_{ACR} hace que $C_{\text{adj}}(\mu,\sigma) < C_{\text{co}}(\mu,\sigma)$, lo que tiene sentido ya que utilizar frecuencias adyacentes no provoca tantas interferencias como usar la misma frecuencia.

Como se puede observar, para cada plan de frecuencias tentativo, el modelo devuelve un valor de coste que indica la calidad de dicho plan atendiendo a la información procedente de la matriz de interferencias. Tanto la información tan precisa de la matriz como los cálculos posteriores del coste de la planificación están motivados por la planificación de frecuencias en redes GSM reales, ya que están relacionados con el cómputo del rendimiento BER (*Bit Error Rate*) de la modulación GMSK (*Gaussian Minimum Shift Keying*) usada en GSM [226].

5.3. Conclusiones

La asignación de frecuencias es una fase fundamental dentro de las redes GSM. Un buen plan de frecuencias permite no sólo aprovechar más eficientemente el espectro de radio disponible, sino también incrementar la calidad del servicio y las ganancias de los operadores. En este capítulo hemos propuesto una nueva formulación matemática para el problema que utiliza información de interferencias muy precisa procedente de redes GSM reales. Esta nueva formulación, junto con los datos de redes GSM que están funcionando ahora mismo en distintas partes del mundo, nos va a permitir enfrentar nuestras propuestas algorítmicas a situaciones reales que aparecen normalmente en las empresas del sector.

Capítulo 6

Optimización del proceso de difusión en MANETs

Con el rápido desarrollo de tecnologías de comunicación inalámbricas así como la proliferación de dispositivos como teléfonos móviles, PDAs y ordenadores portátiles, las redes móviles ad hoc (MANETs) han surgido como un importante campo de investigación dentro de las redes de ordenadores ya que, por un lado, no requieren el soporte de ninguna infraestructura previa y, por otro, se pueden desplegar y utilizar a un coste muy bajo. Las MANETs están compuestas por un conjunto de dispositivos de comunicación llamados *nodos* (o *dispositivos*) que se mueven y organizan de alguna forma arbitraria. La movilidad de estos dispositivos, junto con las conexiones inalámbricas de alcance limitado, hacen que la topología de la red cambie rápidamente y de manera impredecible. Este comportamiento tan dinámico constituye uno de los principales obstáculos para conseguir comunicaciones eficientes.

Este capítulo se centra en el problema de difusión (o *broadcasting*) en MANETs, es decir, el proceso en el que un nodo de la red envía un mensaje a todos los demás nodos que componen dicha red. Si bien la difusión es una operación muy común a nivel de aplicación, también es ampliamente utilizada para resolver varios problemas que se presentan en las propias redes, siendo por ejemplo, el mecanismo básico de muchos mecanismos de encaminamiento de paquetes. En general, en cualquier MANET dada, las operaciones de difusión son muy frecuentes debido a la movilidad de los nodos (por ejemplo, localizar un nodo en particular, enviar una señal de alarma, y/o encontrar un camino hacia un destino). En este tipo de redes, la difusión también se utiliza como último recurso para proporcionar servicios multicast (comunicaciones de un nodo a grupos de nodos). Por tanto, la elección de una estrategia de difusión apropiada va a provocar un incremento en el rendimiento de la red.

Aunque la difusión en sí puede parecer una tarea fácil, esta operación es realmente complicada de realizar de forma eficiente y fiable en el contexto de las MANETs. En este sentido, diferentes grupos de investigación han propuesto estrategias de difusión específicas para MANETs genéricas [232, 256], que están diseñadas para conseguir un rendimiento aceptable en cualquier situación. Sin embargo, existen muchos tipos diferentes de MANETs atendiendo al contexto y al objeto último de su aplicación, por lo que otros autores se han centrado en el desarrollo de estrategias muy especializadas para clases particulares de MANETs. *Inundación con Retrasos y Vecindarios Acumulativos* (o *Delayed Flooding with Cumulative Neighborhood* –DFCN) [112] es un ejemplo de estrategia diseñada específicamente para MANETs *Metropolitanas* [48]. Éste es el tipo de redes que se consideran en este capítulo. Las MANETs metropolitanas se caracterizan por tener una densidad de nodos muy heterogénea y dinámica en las que las zonas de alta densidad no se mantienen activas

durante todo el tiempo. Esto lleva a topologías compuestas por subconjuntos de redes ad hoc que pueden unirse y separarse dinámicamente durante la propia operación de difusión. Ejemplos claros de estas redes son aeropuertos, estaciones de tren o edificios de oficinas.

Supongamos que una cadena de centros comerciales pretende introducir en sus edificios un servicio con el que, la gente que disponga de teléfonos móviles, PDAs, etc., pueda recibir eventualmente pequeños anuncios procedentes de las propias tiendas del edificio en los que publicitan sus productos, ofertas especiales, etc. En este escenario tan concreto sería deseable encontrar la mejor estrategia posible, cualquiera que sea el protocolo de difusión empleado (por ejemplo, DFCN). Éste es el enfoque seguido en este estudio: optimizar el servicio de difusión de mensajes para una red concreta. Ese ajuste óptimo se ha formulado como un problema de optimización en el que varios objetivos han de satisfacerse a la vez. Es lo que se conoce como optimización multiobjetivo (ver Sección 3.3). Objetivos potenciales en este problema son la maximización del número de dispositivos a los que llega el mensaje (cobertura), minimizar la utilización de la red (ancho de banda) y/o minimizar la duración en sí del proceso de difusión. En nuestro caso, la estrategia de broadcasting considerada para su optimización es DFCN y los escenarios objetivo donde se despliega son MANETs metropolitanas. Dado que disponer de este tipo de redes no es fácil, hemos tenido que realizar simulaciones para diseñar y evaluar los escenarios utilizados. Este problema, que se ha definido por primera vez durante el desarrollo de esta tesis, es, hasta donde conocemos, el primer intento en el que el ajuste de una estrategia de difusión para un escenario concreto en MANETs metropolitanas se formula como un problema de optimización.

El resto del capítulo se organiza de la siguiente manera. En la Sección 6.1 se describen las características de la red objetivo que definen nuestro problema de optimización. La Sección 6.2 está dedicada a la presentación de DFCN [112], la estrategia de difusión que tenemos que ajustar. Los problemas de optimización que se definen en este trabajo están incluidos en la Sección 6.3. Finalmente, El capítulo termina con las principales conclusiones obtenidas.

6.1. Redes Móviles Ad Hoc Metropolitanas. El Simulador Madhoc

Las redes móviles ad hoc metropolitanas son MANETs con algunas propiedades particulares. En primer lugar, tienen una o más áreas donde la densidad de los nodos es mayor que la media: se denominan *áreas de alta densidad* y se pueden detectar estadísticamente. Un área de alta densidad puede ser, por ejemplo, un supermercado, un aeropuerto o una oficina. En segundo lugar, las áreas de alta densidad no se mantienen activas todo el tiempo, es decir, pueden aparecer y desaparecer de la red (por ejemplo, un supermercado está abierto de 9 de la mañana a 10 de la noche, y fuera de ese periodo, la densidad de nodos en el área correspondiente es casi cero). En la Figura 6.1 mostramos un ejemplo de red metropolitana de 4 km^2 con 2000 dispositivos.

Debido al coste de desplegar este tipo de redes, la realización de experimentos utilizando redes reales es muy complicado por lo que suele recurrirse a simulaciones software. En este caso hemos utilizado *Madhoc*[1] [112], un simulador de MANETs metropolitanas. Su objetivo es proporcionar una herramienta para simular diferentes niveles de servicios basados en distintas tecnologías de MANETs para distintos entornos.

En el contexto de las MANETs metropolitanas se suelen encontrar varias configuraciones topológicas. Algunos ejemplos son redes construidas espontáneamente por gente moviéndose en lugares concretos, tales como mercados, estaciones de trenes, centros comerciales y barrios de ciudades. Todos estos escenarios tienen varias características distintas, como la movilidad y la densidad de los dispositivos, el tamaño del área y la presencia o no de obstáculos para movilidad y atenúan la

[1] El simulador Madhoc está disponible de manera gratuita en `http://www-lih.univ-lehavre.fr/~hogie/madhoc/`

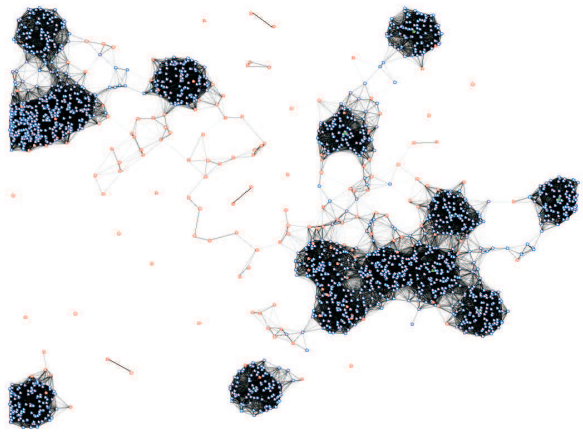


Figura 6.1: Un ejemplo de MANET metropolitana.

fuerza de la señal, entre otras. En este trabajo se utilizan tres escenarios realistas muy distintos, implementados todos ellos en Madhoc. Estos escenarios se corresponden con los siguientes entornos del mundo real: un centro comercial, un área metropolitana y el entorno de una autopista. Su descripción detallada se deja para el Capítulo 10, donde se aborda la resolución del problema.

Para conseguir hacer nuestros estudios más realistas, hemos incluido en las simulaciones una *ventana de observación*, que considera un porcentaje del espacio de simulación de forma que, a la hora de tomar medidas de los dispositivos, sólo se consideran aquellos que están dentro de esta ventana. Esto hace posible la simulación de los nodos entrando y saliendo de la red cuando llegan o abandonan la ventana de observación, respectivamente. Se permite así la existencia de un número variable de dispositivos en la red, como sucede en las MANETs reales. En todas las pruebas en este trabajo, esta ventana de observación cubre el 70 % del área total. Por ejemplo, en la Figura 6.2 podemos observar una MANET simulando un entorno de un centro comercial (a la izquierda) y la ventana de observación que estudiamos (a la derecha); suponiendo el 70 % de la totalidad de la red. Los círculos representan tiendas, mientras que los puntos son dispositivos (los que están fuera de la ventana de observación aparecen en gris más claro, lo que quiere decir que no se consideran al tomar las medidas de la red).

6.2. Inundación con Retrasos y Vecindarios Acumulativos

Williams y Camp [256] y, más recientemente, Stojmenovic y Wu [232] han propuesto dos de los análisis de protocolos de difusión más referenciados. En su trabajo, Stojmenovic y Wu [232] constataron que los protocolos se pueden clasificar según la naturaleza de su algoritmo –determinismo (no hace uso de la aleatoriedad), fiabilidad (garantía de cobertura total)– o según la información requerida para su ejecución (información de red, mensajes de contenido "hola", mensajes de contenido broadcast). De manera similar, Wu y Lou [257] clasifican los protocolos como *centralizados* y *localizados* [199]. Los protocolos centralizados necesitan tener un conocimiento global o casi global de la red, por lo que no son escalables. Los protocolos localizados son aquellos que necesitan conocimiento del vecindario a 1 ó 2 saltos, siendo el vecindario a 1 salto de un dispositivo dado el conjunto de dispositivos visibles directamente por él, y el vecindario a 2 saltos es este vecindario más los vecindarios de sus vecinos.

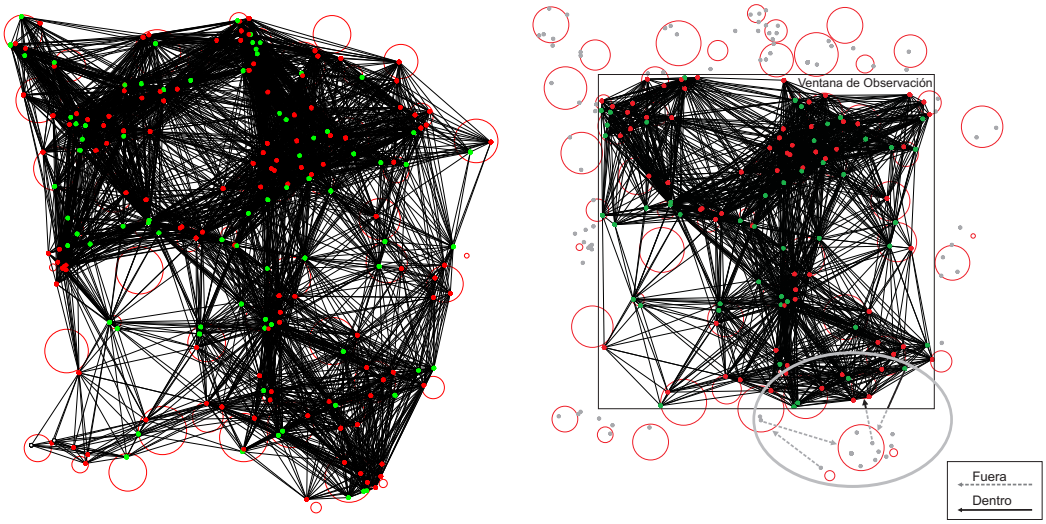


Figura 6.2: Los efectos de introducir una ventana de observación en la MANET estudiada.

Utilizando estas clasificaciones, la inundación con retrasos y vecindarios acumulativos (DFCN) [112] es un algoritmo determinista. No se trata de una nueva aproximación para calcular conjuntos dominantes, sino de un protocolo totalmente localizado que define heurísticas basadas en información en 1 salto. Esto permite a DFCN tener gran escalabilidad. Los mensajes "hola" que intercambian los nodos para conocer su vecindario en todo momento no llevan información adicional. Sólo los mensajes de difusión deben incluir la lista de los nodos vecinos.

Para poder ejecutar el protocolo DFCN, debemos asumir lo siguiente:

- Como muchos otros protocolos de difusión (FWSP, SBA, etc.) [152, 201], DFCN necesita conocer el vecindario de 1-salto para su funcionamiento. Esto se consigue utilizando los paquetes "hola" en una capa de red inferior. El conjunto de vecinos del dispositivo s se llama $N(s)$.

- Cada mensaje m (salvo mensajes tipo hola) contiene el conjunto de identificadores (IDs) de los vecinos de 1 salto de sus remitentes más recientes incluido en su cabecera.

- Cada dispositivo mantiene información local sobre todos los mensajes recibidos. Cada instancia de esta información local está formada por los siguientes puntos:

  - el ID del mensaje recibido;
  - el conjunto de IDs de los dispositivos que se sabe que han recibido el mensaje;
  - la decisión de si el mensaje debe reenviarse o no.

- DFCN necesita utilizar un retraso aleatorio antes de reemitir un mensaje de difusión m. Este retraso, llamado *Retraso de Estimación Aleatoria (RAD o* Random Assessment Delay*)*, se usa para prevenir colisiones. Para ser más exactos, cuando un dispositivo s envía un mensaje m, todos los dispositivos de $N(s)$ lo reciben a la vez. Es probable entonces que todos ellos reenvíen m simultáneamente, y esta simultaneidad implica colisiones de red. El objetivo de RAD es retrasar aleatoriamente la retransmisión de m en cada nodo. Como cada dispositivo de $N(s)$ espera a que termine un RAD distinto antes del reenvío de m, el riesgo de colisiones se reduce considerablemente.

DFCN es un algoritmo orientado a eventos que puede dividirse en tres partes principales: las dos primeras tienen el objetivo de manejar los eventos que llegan, que son (i) recepción de un nuevo mensaje y (ii) detección de un nuevo vecino. La tercera parte (iii) consiste en tomar la decisión de emitir como respuesta a uno de los dos eventos anteriores. El comportamiento resultante de la recepción de un mensaje se denomina comportamiento *reactivo*; en cambio, cuando se descubre un nuevo vecino, el comportamiento seguido se denomina comportamiento *proactivo*.

Consideremos que s_1 y s_2 son dos dispositivos cada uno en el vecindario a 1 salto del otro. Cuando s_1 envía un paquete a s_2, le añade al paquete la información sobre el conjunto $N(s_1)$. Cuando lo recibe, s_2 tiene conocimiento de que cada dispositivo en $N(s_1)$ ha recibido también el paquete. El conjunto de vecinos de s_2 a los que *potencialmente* no le ha llegado todavía dicho paquete es entonces $N(s_2) - N(s_1)$. Si s_2 reemite el paquete, el número máximo de nuevos dispositivos que se alcanzarían está determinado por la función heurística:

$$h(s_2, s_1) = |N(s_2) - N(s_1)| \ . \tag{6.1}$$

Para minimizar el uso de la red causado por el posible reenvío de paquetes, un mensaje será reenviado únicamente si el número de nuevos dispositivos que se alcanzarían es mayor que un umbral dado. Este umbral se establece en función del número de dispositivos en el vecindario (la densidad de red local) del dispositivo receptor s_2, y se escribe "threshold($|N(s)|$)". La decisión que toma s_2 de reenviar el paquete recibido de s_1 se define por una función lógica:

$$B(s_2, s_1) = \begin{cases} \text{verdadero} & \text{si } h(s_2, s_1) \geq \text{threshold}(|N(s_2)|) \\ \text{falso} & \text{en otro caso} \ . \end{cases} \tag{6.2}$$

Si se excede el umbral, el dispositivo receptor s_2 pasa a ser emisor. El mensaje es definitivamente enviado cuando el retraso aleatorio (definido por RAD) termina. La función umbral, que permite a DFCN facilitar el mensaje de re-difusión cuando la conectividad es baja, depende del tamaño del vecindario n, como se muestra en:

$$\text{threshold}(n) = \begin{cases} 1 & \text{si } n \leq \text{densidadDeSeguridad} \\ \text{mínimaGanancia} * n & \text{en otro caso} \ , \end{cases} \tag{6.3}$$

donde *densidadDeSeguridad* es la densidad de seguridad máxima por debajo de la cual DFCN siempre continua con el proceso de difusión y *mínimaGanancia* es un parámetro de DFCN utilizado para calcular el umbral mínimo necesario para enviar un mensaje, es decir, la proporción entre el número de vecinos que no han recibido el mensaje y el número total de vecinos.

Cada vez que un dispositivo s descubre un nuevo vecino, el RAD de este dispositivo para todos los mensajes se pone a cero y, por tanto, los mensajes son inmediatamente candidatos para ser transmitidos. Si $N(s)$ es mayor que un umbral dado, llamado *proD*, este comportamiento se desactiva, por lo que no se podrá realizar ninguna acción cuando se detecta un nuevo vecino.

6.3. Definición de los problemas de optimización

En esta sección se formulan dos problemas multiobjetivo que optimizan el funcionamiento del protocolo de difusión DFCN en MANETs metropolitanas. De la descripción de DFCN proporcionada en la sección anterior, se identifican los siguientes parámetros que deben ser ajustados:

mínimaGanancia es la ganancia mínima que debe alcanzar un nodo para reemitir un mensaje. Éste es el parámetro más importante para ajustar DFCN, ya que minimizar el ancho de banda

va a depender mucho de la densidad de la red. Los valores que puede tomar este parámetro varían entre 0,0 y 1,0.

[límiteInferiorDeRAD, límiteSuperiorDeRAD] definen el intervalo de valores que puede tomar RAD (retraso aleatorio para re-difusión). Ambos parámetros toman valores del intervalo [0,0, 10,0] (en milisegundos).

proD es la densidad máxima ($proD \in [0, 100]$) con la que todavía se necesita usar el comportamiento proactivo (i.e., reaccionar ante la detección de nuevos vecinos) para complementar el comportamiento reactivo.

densidadDeSeguridad define una densidad máxima de seguridad del umbral "threshold(n)", que varía entre 0 y 100 dispositivos.

Estos cinco parámetros o cinco variables de decisión corresponden a una configuración de DFCN y caracterizan el espacio de búsqueda de nuestro problema de optimización. Hemos establecido intervalos suficientemente amplios para los valores de estos parámetros, ya que queremos incluir todas las posibilidades razonables que se pueden encontrar en el escenario real. En todo proceso de difusión hay tres objetivos, claramente contrapuestos, que deben satisfacerse a la vez (ver Ecuación 6.4): minimizar la duración del propio proceso de difusión, maximizar la cobertura que alcanzan los mensajes y minimizar el número de transmisiones empleadas (ancho de banda). Hemos elegido, por tanto, una formulación multiobjetivo de nuestro problema de optimización, al que hemos denominado DFCNT (*DFCN Tuning* o ajuste de DFCN).

$$DFCNT = \begin{cases} f_1(x) = \text{duración del proceso de difusión} & -\text{minimizar} \\ f_2(x) = \text{cobertura} & -\text{maximizar} \\ f_3(x) = \text{número de transmisiones} & -\text{minimizar} \end{cases} \tag{6.4}$$

Además, en este contexto también es interesante considerar la cobertura de la red como una restricción que ha de cumplirse en vez de como un objetivo. Ya que, por ejemplo, configuraciones de DFCN que no alcancen, al menos, un 90 % de los nodos que componen la red, podrían no ser lo suficientemente aceptable. Lo que interesa, entonces, es encontrar la relación entre el ancho de banda que necesita el protocolo para funcionar y la duración de cada operación de difusión. El problema resultante, al que hemos llamado cDFCNT (*constrained DFCNT* o DFCNT*con restricción*), es un problema bi-objetivo con una restricción.

$$cDFCNT \begin{cases} f_1(x) = \text{duración del proceso de difusión} & -\text{minimizar} \\ f_2(x) = \text{número de transmisiones} & -\text{minimizar} \\ g(x) \ = \text{cobertura} \geq 90\,\% & -\text{restricción} \end{cases} \tag{6.5}$$

6.4. Conclusiones

Este capítulo presenta una aproximación al problema de ajuste óptimo de DFCN, un protocolo de difusión para MANETs. La idea es obtener la mejor configuración posible de DFCN para un escenario concreto, de forma que se obtenga una estrategia óptima. Se ha planteado el problema desde un enfoque multiobjetivo donde hay que minimizar la duración del proceso de difusión, maximizar la cobertura de la red y minimizar el uso de la red. De hecho, se han definido dos MOPs, uno con los tres objetivos anteriores y otro con dos objetivos y la cobertura como restricción. El problema es muy novedoso y prometedor, ya que abre una línea clara de investigación en optimización para MANETs en un campo que está en pleno auge.

Parte III

Diseño y evaluación de metaheurísticas para los problemas propuestos

Capítulo 7

Propuestas metodológicas

Tras el repaso por las metaheurísticas en el Capítulo 3, aquí presentaremos de forma detallada los algoritmos utilizados para abordar los tres problemas de redes de telecomunicaciones planteados. Con la intención de presentar un trabajo coherente, hemos seleccionado dos tipos de metaheurísticas con las que hemos resuelto los tres problemas considerados, Algoritmos Evolutivos (EAs) y Búsqueda Dispersa (SS). Dado que tenemos problemas tanto mono como multiobjetivo, hemos desarrollado versiones de los algoritmos para trabajar con ambos tipos de optimización.

En cuanto a los EAs (Sección 7.1), hemos utilizado la subfamilia de los algoritmos genéticos (GAs) y, más concretamente, los algoritmos genéticos con selección por estado estacionario (*steady state GAs*). La elección de este tipo de algoritmos se ha fundamentado en dos razones básicas. Primero, son técnicas muy conocidas y que proporcionan soluciones muy precisas para una gran cantidad de problemas de optimización, y, segundo, las características de la selección por estado estacionario permiten que el modelo se pueda extender para su ejecución en sistemas grid y así poder aprovechar la capacidad de cómputo que éstos son capaces de proporcionar. Esto es especialmente necesario para resolver problemas del mundo real como los tratado en esta tesis.

Aunque las bases de la búsqueda dispersa se plantearon en la década de los 70, esta técnica ha comenzado a utilizarse ampliamente en los últimos años, donde ha mostrado ser muy eficiente para un gran número de problemas de optimización. Su elección se debe a que su aplicación a los problemas considerados es una contribución como tal, ya que, hasta donde conocemos, nunca antes se han habían resuelto con búsqueda dispersa.

Por último, atendiendo al contexto y a las particularidades de cada uno de los tres problemas planteados, hemos utilizado en cada caso una metaheurística que consideramos como muy adecuada para su resolución. Así, para el primer problema abordado, la planificación de celdas en redes de telefonía, hemos planteado un modelo paralelo basado en PAES (*Pareto Archived Evolution Strategy*) [136], ya que los tiempos de cómputo para evaluar las soluciones tentativas son muy elevados. En esta búsqueda por la máxima eficiencia, decidimos utilizar PAES porque se considera como la estrategia más simple capaz de generar una aproximación al frente de Pareto de problemas multiobjetivo. En el caso de la planificación de frecuencias, la propuesta algorítmica es un ACO, ya que el problema se puede plantear como un problema de búsqueda en caminos mínimos en grafos y es bien conocido que este tipo de algoritmos funcionan con grandes niveles de efectividad en este contexto. A pesar de este hecho, los trabajos relacionados son escasos y existen multitud de oportunidades de investigación aquí. La última propuesta que presentamos en este capítulo es un algoritmo genético celular para la optimización de la estrategia de difusión en MANETs. La robustez de este tipo de algoritmos para explorar en espacios de búsqueda complejos fue la principal motivación. Además, estas propuestas son los primeros GAs celulares para optimización multiobjetivo que siguen el modelo canónico de dichos algoritmos. En todos los casos, damos sólo

los esquemas generales de cada técnica puesto que, al ser aplicadas a problemas tan distintos, los operadores que se utilizan son diferentes, por lo que los detalles de cada uno se explican más adelante.

La estructura del capítulo es como sigue. A continuación se introducen los algoritmos genéticos, particularizando en las dos aproximaciones mono y multiobjetivo utilizadas: ssGA y ssNSGA-II. La Sección 7.2 está dedicada a los algoritmos de búsqueda dispersa usadas en la tesis. Los algoritmos específicos utilizados para cada problema se incluyen en la Sección 7.3. El capítulo termina con las principales conclusiones alcanzadas.

7.1. Algoritmos evolutivos

Alrededor de los años 60, algunos investigadores visionarios coincidieron, de forma independiente, en la idea de implementar algoritmos basados en el modelo de evolución orgánica como un intento de resolver tareas complejas de optimización en ordenadores. Hoy en día, debido a su robustez, a su amplia aplicabilidad, y también a la disponibilidad de una cada vez mayor potencia computacional, e incluso programas paralelos, el campo de investigación resultante, el de la computación evolutiva [20], recibe una atención creciente por parte de los investigadores de un gran número de disciplinas.

El marco de la computación evolutiva establece una aproximación para resolver el problema de buscar valores óptimos mediante el uso de modelos computacionales basados en procesos evolutivos. Los EAs son técnicas de optimización que trabajan sobre poblaciones de soluciones y que están diseñadas para buscar valores óptimos en espacios complejos. Están basados en procesos biológicos que se pueden apreciar en la naturaleza, como la selección natural [61] o la herencia genética [174]. Parte de la evolución está determinada por la selección natural de individuos diferentes compitiendo por recursos en su entorno. Algunos individuos son mejores que otros y es deseable que aquellos individuos que son mejores sobrevivan y propaguen su material genético.

La reproducción sexual permite el intercambio del material genético de los cromosomas, produciendo así descendientes que contienen una combinación de la información genética de sus padres. Éste es el *operador de recombinación* utilizado en los EA, también llamado *operador de cruce*. La recombinación ocurre en un entorno en el que la selección de los individuos que tienen que emparejarse depende, principalmente, del valor de la función de *fitness* del individuo, es decir, de cómo de bueno es el individuo comparado con los de su entorno.

Como en el caso biológico, los individuos pueden sufrir mutaciones ocasionalmente (*operador de mutación*). La mutación es una fuente importante de diversidad para los EAs. En un EA, se introduce normalmente una gran cantidad de diversidad al comienzo del algoritmo mediante la generación de una población de individuos aleatorios. La importancia de la mutación, que introduce aún más diversidad mientras el algoritmo se ejecuta, es objeto de debate. Algunos se refieren a la mutación como un operador de segundo plano, que simplemente reemplaza parte de la diversidad original que se haya podido perder a lo largo de la evolución, mientras que otros ven la mutación como el operador que juega el papel principal en el proceso evolutivo.

En la Figura 7.1 se muestra el esquema de funcionamiento de un EA típico. Como puede verse, un EA procede de forma iterativa mediante la evolución de los individuos pertenecientes a la población actual. Esta evolución es normalmente consecuencia de la aplicación de operadores estocásticos de variación sobre la población, como la selección, recombinación y mutación, con el fin de calcular una generación completa de nuevos individuos. El criterio de terminación consiste normalmente en alcanzar un número máximo de iteraciones (programado previamente) del algoritmo, o encontrar la solución óptima al problema (o una aproximación a la misma) en caso de que se conozca de antemano.

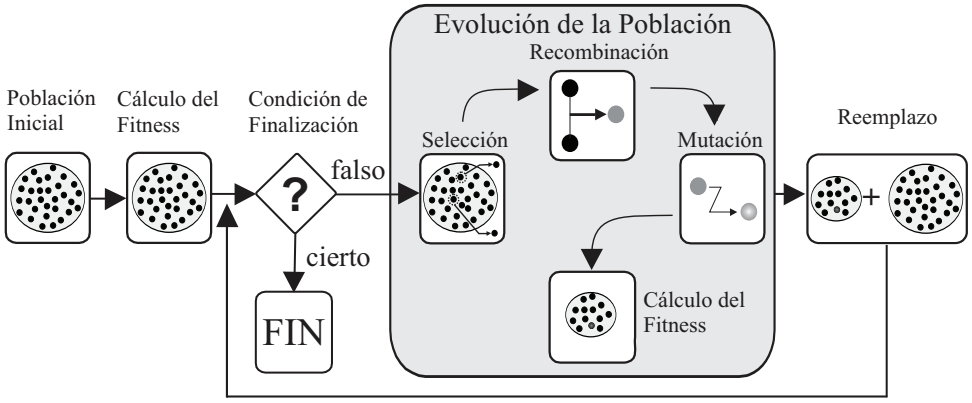


Figura 7.1: Funcionamiento de un EA canónico.

Pasaremos ahora a analizar detalladamente el funcionamiento de un algoritmo evolutivo, cuyo pseudocódigo se muestra en el Algoritmo 7.1. Como se ha dicho ya con anterioridad, los algoritmos evolutivos trabajan sobre poblaciones de individuos, que son soluciones tentativas del problema. La población inicial está compuesta usualmente por individuos creados aleatoriamente, aunque también existe cierta tradición en el uso de técnicas de optimización (preferiblemente de poca carga computacional) para crear los individuos que formarán la población inicial, permitiendo así que el EA comience su ejecución sobre un conjunto de soluciones más prometedoras que las generadas aleatoriamente. Tras la generación de la población inicial, se calcula el valor de adecuación (*fitness*) de cada uno de los individuos que la forman y el algoritmo entra en el bucle reproductor. Este bucle consiste en la generación de una nueva población mediante la selección de padres, la recombinación de éstos y la mutación de los descendientes obtenidos. Tras este proceso, los individuos son evaluados. Esta nueva población generada por el bucle reproductor (P') se utilizará, junto con la población actual (P), para obtener la nueva población de individuos de la siguiente generación. Al final, el algoritmo devolverá la mejor solución encontrada durante la ejecución.

Como puede verse, el algoritmo comprende las tres fases principales: selección, reproducción y reemplazo. A continuación detallamos estas tres fases:

- **Selección:** partiendo de la población inicial P de μ individuos, se crea una nueva población temporal (P') de λ individuos. Generalmente los individuos más aptos (aquellos correspondientes a las mejores soluciones contenidas en la población) tienen un mayor número de

Algoritmo 7.1 Pseudocódigo de un algoritmo evolutivo.

```
1: P ← generarPoblaciónInicial();
2: evaluar(P);
3: while not condiciónParada() do
4:    P' ← seleccionarPadres(P);
5:    P' ← aplicarOperadoresDeVariación(P');
6:    evaluar(P');
7:    P ← seleccionarNuevaPoblación(P, P');
8: end while
9: return la mejor solución encontrada
```

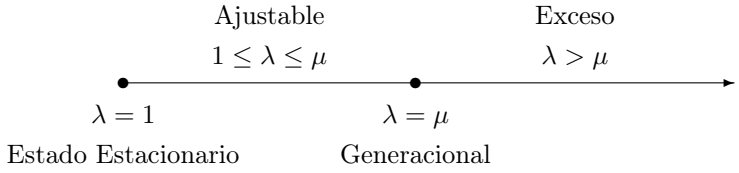


Figura 7.2: Diferencia entre los diversos esquemas de selección.

instancias que aquellos que tienen menos aptitud (selección natural). De acuerdo con los valores de μ y λ podemos definir distintos esquemas de selección (Figura 7.2):

1. **Selección por estado estacionario.** Cuando $\lambda = 1$ tenemos una selección por estado estacionario (*steady-state*) en la que únicamente se genera un hijo en cada paso de la evolución.
2. **Selección generacional.** Cuando $\lambda = \mu$ tenemos una selección por generaciones en la que genera una nueva población completa de individuos en cada paso.
3. **Selección ajustable.** Cuando $1 \leq \lambda \leq \mu$ tenemos una selección intermedia en la que se calcula un número ajustable (*generation gap*) de individuos en cada paso de la evolución. Los anteriores son casos particulares de éste.
4. **Selección por exceso.** Cuando $\lambda > \mu$ tenemos una selección por exceso típica de los procesos naturales reales.

- **Reproducción:** en esta fase se aplican los operadores reproductivos a los individuos de la población P' para producir una nueva población. Típicamente, esos operadores se corresponden con la recombinación de parejas y con la mutación de los nuevos individuos generados. Estos operadores de variación son, en general, no deterministas, es decir, no siempre se tienen que aplicar a todos los individuos y en todas las generaciones del algoritmo, sino que su comportamiento viene determinado por su probabilidad asociada.
- **Reemplazo:** finalmente, los individuos de la población original son sustituidos por los individuos recién creados. Este reemplazo usualmente intenta mantener los mejores individuos eliminando los peores. Dependiendo de si, para realizar el reemplazo, se tienen en cuenta la antigua población P podemos obtener dos tipos de estrategia de reemplazo:

  1. (μ, λ) si el reemplazo se realiza utilizando únicamente los individuos de la nueva población P'. Se debe cumplir que $\mu \leq \lambda$
  2. $(\mu + \lambda)$ si el reemplazo se realiza seleccionando μ individuos de la unión de P y P'.

Estos algoritmos establecen un equilibrio entre la explotación de buenas soluciones (fase de selección) y la exploración de nuevas zonas del espacio de búsqueda (fase de reproducción), basado en el hecho de que la política de reemplazo permite la aceptación de nuevas soluciones que no mejoran necesariamente las existentes.

Fueron cuatro los primeros tipos de algoritmos evolutivos que surgieron [121]. Estas cuatro familias de algoritmos fueron desarrolladas simultáneamente por distintos grupos de investigación. Los *algoritmos genéticos* (GA), fueron inicialmente estudiados por Holland [113, 114], en Ann Arbor (EEUU), Bremermann [32] en Berkeley (EEUU), y Fraser [91] en Sidney (Australia). Las *estrategias evolutivas* fueron propuestas por Rechenberg [212, 213] y Schwefel [224] en Berlin (Alemania),

mientras que la *programación evolutiva* se propuso por primera vez por Fogel [85] en San Diego (California). Por último, la cuarta familia de algoritmos, la *programación genética*, surgió dos décadas más tarde, en 1985, como una adaptación, hecha por Cramer [53], de un algoritmo genético que trabajaba con genes en forma de árbol, además de las cadenas de caracteres binarios utilizadas tradicionalmente en GA.

La primera propuesta algorítmica para resolver los tres problemas de telecomunicaciones descritos en esta tesis son los algoritmos genéticos. En concreto, nos hemos centrado en los algoritmos genéticos de estado estacionario. Esta decisión tiene que ver con dos características fundamentales de este tipo de algoritmos que los hacen más adecuados que los GAs generacionales para el contexto en el que nos movemos. Por una parte, es bien conocido dentro del campo de los GAs que la selección por estado estacionario permite al algoritmo converger más rápidamente que el esquema generacional [220, 221]. De hecho, en [221] se llega a demostrar el GA estado estacionario evoluciona con la misma dinámica que un GA generacional utilizando la mitad del coste computacional. Como estamos abordando problemas del mundo real que conllevan tiempos de evaluación muy altos para la función de fitness, este esquema de selección va a permitir reducir considerablemente los tiempos de cómputo. Por otra parte, este esquema es el más adecuado para la posterior extensión grid del algoritmo debido a características asíncronas propias de la estrategia (ver Sección 7.4). Puesto que vamos a tratar con problemas tanto mono como multiobjetivo, hemos utilizado dos GAs estado estacionario distintos que se describen en las dos secciones siguientes. Es importante mencionar aquí que debido a la complejidad de los problemas abordados, tanto la representación como los operadores genéticos utilizados dependen de éstos, por lo que sólo incluimos los esquemas generales de los dos algoritmos.

7.1.1. ssGA

El algoritmo ssGA (*steady state GA*) utilizado es un algoritmo genético estado estacionario estándar que utiliza selección por torneo binario (Algoritmo 7.2). La codificación de soluciones, la recombinación y la mutación dependerán del problema concreto. Como se puede observar en la descripción del algoritmo, hemos incorporado una fase de búsqueda local (línea 7) por lo que ssGA puede considerarse como un algoritmo híbrido [235]. Esta modificación está motivada el problema de la asignación de frecuencias, donde la búsqueda local es imprescindible para obtener soluciones de calidad y, a la vez, permite realizar computación intensiva sobre las soluciones. Esto, que en un principio puede ser un inconveniente a la hora de abordar los problemas, nos permite extender ssGA de forma directa para su ejecución en sistemas de computación grid, y así aprovechar de forma eficiente una gran cantidad de recursos computacionales con el objetivo de resolver problemas del mundo real.

Algoritmo 7.2 Pseudocódigo de ssGA.

```
1: población ← ∅
2: inicializar(población)
3: while no se cumple la condición de terminación do
4:    padres ← selección(población)
5:    hijo ← recombinación(padres,p_c)
6:    hijo ← mutación(hijo,p_m)
7:    hijo ← búsquedaLocal(hijo,pasosDeBúsquedaLocal)
8:    hijo ← evaluar(hijo)
9:    población ← insertar(población,hijo)
10: end while
```

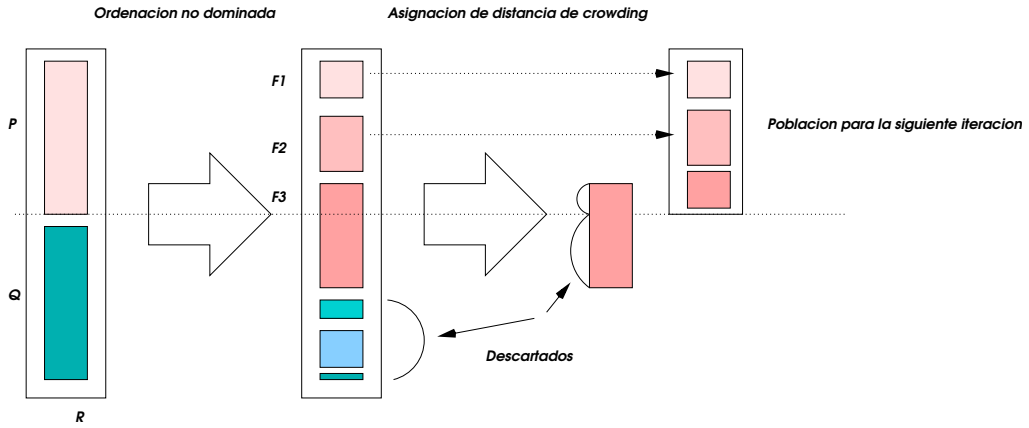


Figura 7.3: Funcionamiento de NSGA-II.

7.1.2. ssNSGA-II

Nuestra propuesta de GA de estado estacionario para optimización multiobjetivo ha consistido en incorporar este esquema de selección a NSGA-II [68], el algoritmos más popular en el campo, donde se considera además como el algoritmo de referencia. Propuesto inicialmente en [67], está basado en la ordenación no dominada de individuos, consistente en dividir una población en diferentes frentes, la primera de ellas formada por individuos a los que no domina nadie, la segunda de ellas formada por individuos que sólo son dominados por un individuo (perteneciente a la primera frontera), y así sucesivamente. Claramente, el número de fronteras será $n \leq N$, donde N es el tamaño de la población que se quiere ordenar. El número de fronteras será $n = N$ en el caso de una población en la que exista una ordenación plana de individuos, es decir, una población en la que es cierta la afirmación de que existe un individuo que domina a todos los demás, y si este individuo es eliminado de la población, esta afirmación se sigue cumpliendo.

Inicialmente, NSGA-II comienza con una población de N individuos creados de forma aleatoria. La población es entonces ordenada basándonos en la dominancia de Pareto. En esta ordenación, a cada uno de los individuos se le asigna un valor de fitness (o rango), que es el valor que se corresponde con su nivel de dominancia. Mediante torneo binario, recombinación y mutación se crea una población de descendientes también de tamaño N. Una vez creada esta población de descendientes, las distintas iteraciones del algoritmo funcionan de la misma forma. Consideremos que P es la población actual y Q es la población de descendientes. Se construye la población $R = P \cup Q$, que es ordenada mediante el algoritmo de ordenación rápida. A partir de R obtenemos los distintos frentes F_1,F_2, etc., donde tenemos que $|F_1 \cup F_2 \cup ... \cup F_l| = |R| = 2N$. Una vez hecho esto, se actualiza la población P. Para ello, se van cogiendo frentes F_i mientras que $|P \cup F_i| \leq N$. Aquí puede ocurrir que el último frente considerado no quepa de forma integra en P. En este caso, se usa una distancia de crowding para ordenar de forma parcial los elementos de dicho frente y escoger los mejores (aquellos que presentan más diversidad, es decir, mayor distancia). La iteración concluye generando una nueva población de descendientes.

Nuestra versión estado estacionario, llamada ssNSGA-II (*steady state NSGA-II*), se diferencia en que cada iteración genera un único individuo ($|Q| = 1$) seleccionado por torneo binario que reemplaza al individuo de peor rango y peor distancia de crowding de la población actual. Este esquema tan simple permite intensificar la búsqueda por aquellas zonas del espacio donde están las soluciones del frente de Pareto, además de incorporar la asincronía necesaria para su posterior extensión grid.

7.2. Búsqueda dispersa

La búsqueda dispersa (*Scatter Search*) [98, 97, 99] es la segunda propuesta algorítmica utilizada para resolver los tres problemas de telecomunicaciones abordados en esta tesis. Decidimos usar esta metaheurística ya que es una técnica con un gran auge en lo últimos años, muy eficiente en determinados casos y, además, nunca antes ha sido aplicada a los problemas considerados. También hay que indicar que para las formulaciones multiobjetivo, hemos diseñado una nueva aproximación, llamada AbYSS (*Archive-based hYbrid Scatter Search*) [192], que forma parte del estado del arte para los bancos de prueba existentes en optimización continua multiobjetivo y, por tanto, su aplicación a problemas del mundo real como los que aquí se tratan es de gran interés.

La búsqueda dispersa es una metaheurística introducida en la década de los setenta [95]. Sus fundamentos están basados en las estrategias de combinar reglas de decisión, especialmente en problemas de secuenciación, así como en la combinación de restricciones (como el método de las restricciones subrogadas). Su funcionamiento se basa en la utilización de una pequeña población, conocida como conjunto de referencia ($RefSet$), cuyos elementos se combinan de forma sistemática para la creación de nuevas soluciones. Además, estas nuevas soluciones pueden pasar por una fase de mejora consistente en la aplicación de una búsqueda local. El conjunto de referencia es inicializado a partir de una población inicial, P, compuesta por soluciones aleatorias lo más dispersas posibles, y es actualizado con las soluciones resultantes de la fase de mejora. Muchas implementaciones de algoritmos de búsqueda dispersa toman como referencia la plantilla propuesta por Glover [97] (ver Figura 7.4), que consiste en definir cinco métodos:

- Generación de soluciones diversas. El método se basa en generar P soluciones diversas del que extraeremos un subconjunto pequeño que se denomina $RefSet$.
- Mejora. Se trata típicamente de un método de búsqueda local para mejorar las soluciones.
- Actualización del conjunto de referencia. Este método se encarga tanto de la generación inicial del conjunto de referencia como de su actualización posterior. El $RefSet$ almacena soluciones en base a dos criterios: calidad y diversidad. Se distingue, por tanto, el $RefSet_1$ y el $RefSet_2$ que almacenan soluciones de alta calidad y soluciones muy diversas, respectivamente.
- Generación de subconjuntos. Es un método para generar subconjuntos de soluciones del $RefSet$ a los que se le aplicará el método de combinación. SS se basa en examinar de forma exhaustiva todas las posibles combinaciones del $RefSet$. La forma más usual consiste en generar parejas de soluciones.
- Combinación de soluciones. Es el método encargado de combinar de alguna manera los subconjuntos de soluciones resultantes del método anterior para generar nuevas soluciones.

En general, la búsqueda dispersa evita el uso de componentes aleatorios, y los operadores típicos de los algoritmos evolutivos como las operaciones de cruce y mutación no se adaptan, teóricamente, a la filosofía de esta técnica. No obstante, existen trabajos tanto en el campo monoobjetivo [110] como en el multiobjetivo [192] que demuestran que la utilización de operadores estocásticos dentro del esquema de SS permite calcular soluciones más precisas. Este enfoque nos va a permitir, además, utilizar los mismos operadores que los GAs presentados en la sección anterior y poder comparar así la eficiencia y efectividad de cada uno de los dos motores de búsqueda, ssGA y SS. Dado que cada problema considerado tendrá sus operadores concretos, a continuación mostramos cómo se interrelacionan estos métodos para definir el procedimiento general de búsqueda dispersa. Al igual que en el caso de los GAs de estado estacionario, se presentan esquemas de SS para optimización mono y multiobjetivo, ya que tenemos formulaciones de ambos tipos para los problemas abordados.

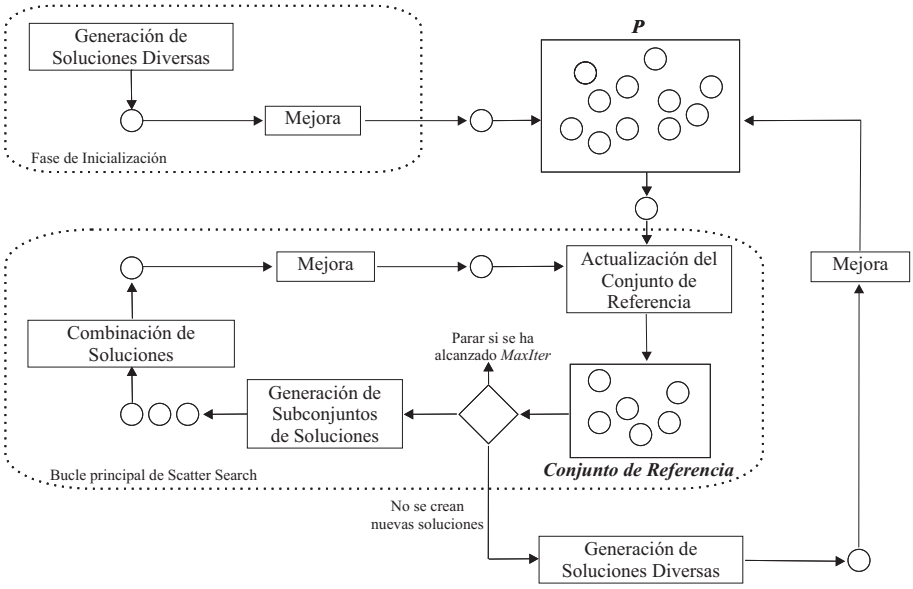


Figura 7.4: Esquema general de la búsqueda dispersa.

7.2.1. SS

El algoritmo de búsqueda dispersa (Figura 7.4) comienza creando un conjunto inicial de soluciones diversas en la fase de inicialización. Esta fase consiste en generar, de forma iterativa, nuevas soluciones mediante la invocación del método de generación de soluciones diversas; cada nueva solución se pasa al método de mejora, que generalmente consiste en una búsqueda local, y la solución resultante se inserta en la población inicial. Después de esta fase de inicialización, comienza el bucle principal de la búsqueda dispersa.

En el bucle principal, primero se construye el conjunto de referencia a partir del conjunto inicial de soluciones usando el método de actualización del conjunto de referencia. Después, las soluciones en el conjunto de referencia se agrupan sistemáticamente en subconjuntos de a dos mediante el método de generación de subconjuntos. En el siguiente paso, las soluciones de cada subconjunto son combinadas de alguna manera para producir nuevos individuos. La forma de combinar dichas soluciones la define el método de combinación. El método de mejora se aplica a cada nueva solución generada y, finalmente, se decide cuales de ellas se insertan en el conjunto de referencia. Este bucle se ejecuta hasta que la condición de terminación sea alcanzada (por ejemplo, un número máximo de iteraciones o no se producen nuevos subconjuntos mediante el método de generación de subconjuntos). Opcionalmente, puede darse un proceso de reinicio. La idea es crear un nuevo conjunto inicial de soluciones que contenga, de partida, las mejores soluciones que se encuentran en el conjunto de referencia, es decir, el $RefSet_1$, mientras que los restantes individuos son generados mediante el método de generación de soluciones diversas.

Como se ha comentado anteriormente, la utilización de esta técnica ya es una contribución de este trabajo porque aún no se ha utilizado para resolver los problemas que estamos considerando. Los métodos particulares dependerán de cada problema y se detallan en los capítulos correspondientes incluidos más adelante.

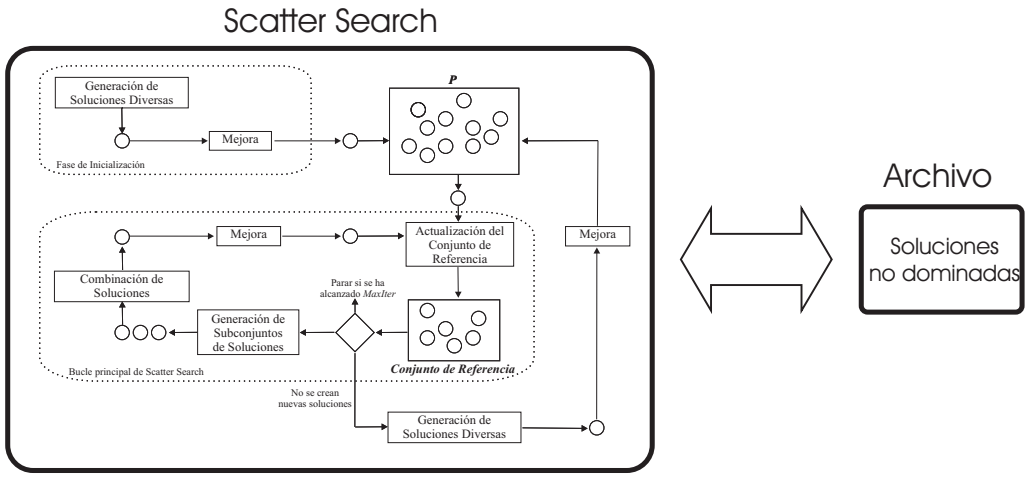


Figura 7.5: Estructura de AbYSS.

7.2.2. AbYSS

Si bien la búsqueda dispersa ha sido utilizada con éxito para resolver una gran variedad de problemas de optimización monoobjetivo, su aplicación a problemas multiobjetivo es más bien marginal. De hecho, las primeras aproximaciones estaban basadas en resolver MOPs con la búsqueda dispersa estándar monoobjetivo (e.g., utilizando técnicas de agregación) [49, 140, 170, 209], aunque también existen propuestas que utilizan optimalidad de Pareto [25, 182, 204, 60, 244]. Nuestra propuesta, denominada AbYSS (*Archive-based hYbrid Scatter Search*) [192], que cae también en esta segunda categoría, combina ideas de tres EAs del estado del arte para resolver MOPs. Por un lado, usa un archivo externo (ver Figura 7.5) para almacenar las soluciones no dominadas que van surgiendo durante la búsqueda, siguiendo el esquema de PAES (*Pareto Archived Evolution Strategy*) [136], pero utilizando la distancia de crowding de NSGA-II [68] como medida de diversidad (*niching*) en lugar de usar el grid adaptativo [136]; por otra parte, la selección de soluciones del conjunto inicial P para construir el conjunto de referencia utiliza la estimación de densidad de SPEA2 [262].

El esquema de funcionamiento de AbYSS sigue la plantilla general de la búsqueda dispersa: inicialmente, el método de generación de soluciones diversas es invocado para generar s soluciones iniciales, cada una de las cuales se pasa al método de mejora. Como resultado se obtiene el conjunto inicial P. Tras esta fase inicial, se realiza un determinado número de iteraciones en las que, en cada una de ellas, se crea el conjunto de referencia, se invoca al método de generación de subconjuntos, y se ejecuta el bucle principal de la búsqueda dispersa hasta que el método de generación de subconjuntos no genere nuevos subconjuntos de soluciones.

Después de un estudio sistemático de las diferentes opciones de diseño de AbYSS [192], la mejor configuración resultante del algoritmo incorpora un par de características que diferencian su bucle principal respecto de la SS estándar. En primer lugar, la generación de subconjuntos no es sistemática, sino que sólo genera subconjuntos de soluciones pertenecientes al $RefSet_1$ por un lado y al $RefSet_2$ por otro. Es decir, no hay combinaciones que contienen soluciones de ambos subconjuntos. Hemos comprobado experimentalmente que esto beneficia la capacidad de búsqueda del algoritmo. Por otro lado, durante la iteración de este bucle principal se van generando soluciones no dominadas que van siendo incorporadas al archivo externo. La cuestión clave en la gestión del archivo es decidir cuándo una solución se debe añadir al archivo. Así, cada vez que llega una solución procedente del bucle interno del método de búsqueda dispersa, se compara una por una

con las que ya pertenecen al archivo. Si la nueva solución es dominada por algún individuo en el mismo (es decir la solución está dominada por el archivo), entonces se descarta; en otro caso, la solución es almacenada en el archivo y se eliminan aquellas dominadas por ésta. Además, si el archivo alcanza su tamaño máximo después de añadir una nueva solución, hay que decidir qué solución debe ser suprimida. Existen diversas estrategias propuestas en la literatura, como el grid adaptativo utilizado en PAES [136]. Aquí, sin embargo, utilizamos la distancia de crowding del NSGA-II [68].

Es importante destacar en este punto que podríamos haber usado el estimador de densidad de SPEA2, como hicimos para el método de actualización del conjunto de referencia. No obstante, hemos decidido utilizar dos estimadores de densidad distintos intentando conseguir una mejor distribución de soluciones no dominadas en el frente de Pareto. El fundamento de esta decisión es que así las soluciones no dominadas han de pasar dos filtros: en primer lugar, las soluciones no pertenecen a la región más poblada de acuerdo con la distancia de crowding y, en segundo lugar, se obtienen a partir de las mejores soluciones del conjunto inicial teniendo en cuenta el estimador de densidad de SPEA2. Resultados experimentales demuestran que la combinación de ambos estimadores obtienen mejor diversidad en los frentes que usar de sólo uno.

Una vez acabado el bucle principal hay una fase de reinicio que consiste en tres pasos. Primero, los individuos en $RefSet_1$ se insertan en P; segundo, los n mejores individuos del archivo externo, de acuerdo a su distancia de crowding, se mueven también a P; y, tercero, se usan el método de generación de soluciones diversas y método de mejora para producir nuevas soluciones hasta completar P.

La idea de mover n individuos desde el archivo al conjunto inicial es intensificar la búsqueda del algoritmo sobre el frente de Pareto encontrado hasta ese momento. El grado de intensificación puede variar dependiendo del número de individuos elegidos. AbYSS usa un valor de n que es el mínimo entre el tamaño del archivo y la mitad del tamaño de P. La condición de parada del algoritmo es la realización de un número máximo de evaluaciones.

De los cinco métodos que requiere la plantilla de búsqueda dispersa para su funcionamiento, sólo el método de generación de subconjuntos ha quedado totalmente definido aquí, ya que los métodos de generación de soluciones diversas, de mejora y de combinación de soluciones son específicos para cada problema. Por tanto, todos se describen más adelante en los capítulos correspondientes a la aplicación a los problemas considerados. No obstante, AbYSS tiene que trabajar con soluciones no dominadas, lo que conlleva algunas modificaciones relacionadas con el método de actualización del conjunto de referencia. Estas modificaciones se detallan a continuación.

El conjunto de referencia es una colección que contiene tanto soluciones de alta calidad como soluciones diversas que son utilizadas para generar otras nuevas. El $RefSet$ está compuesto a su vez por dos subconjuntos, $RefSet_1$ y $RefSet_2$, de tamaño p y q, respectivamente. El primero de ellos contiene las soluciones de mejor calidad en P, mientras el segundo de ellos incluye aquellas soluciones que incorporan diversidad al conjunto. Siguiendo el esquema propuesto en [182], AbYSS construye el $RefSet_2$ seleccionando aquellos individuos de P cuya mínima distancia euclídea al $RefSet_1$ sea máxima. Sin embargo, para crear el conjunto $RefSet_1$ hemos tenido que redefinir el concepto de "mejor individuo".

Como se dijo anteriormente, este método se usa además para actualizar el conjunto de referencia con las nuevas soluciones obtenidas durante el bucle principal de la búsqueda dispersa. Así, para seleccionar los p mejores individuos de P, AbYSS utiliza la estrategia de SPEA2, es decir, a los individuos en P se les asigna un valor de fitness que será la suma del strength raw fitness y un valor de estimación de densidad de soluciones. Mientras que el valor de strength de un individuo es el número de soluciones que este domina en una población, su strength raw fitness es la suma de los strengths de todos los individuos que dominan a éste. Por su parte, la estimación de densidad se basa en calcular la distancia al k-ésimo individuo más cercano en la población (más detalles sobre esto se pueden encontrar [262]).

Una vez construido el conjunto de referencia, sus soluciones se combinan para generar nuevos individuos que, al acabar dicha fase, son pasados al procedimiento de mejora. Después, hay que comprobar si estas nuevas soluciones han de ser incluidas en el conjunto de referencia. Así , una nueva solución se convierte en miembro del conjunto de referencia si se cumple alguna de las siguientes condiciones:

- El nuevo individuo es mejor respecto al valor de la función objetivo que el peor individuo del $RefSet_1$.
- El nuevo individuo tiene un mejor (mayor) valor de distancia al conjunto de referencia que el individuo con peor valor (más bajo) de distancia en $RefSet_2$.

Si bien la segunda condición es válida tanto para mono como para multiobjetivo, la primera de las condiciones supone redefinir de nuevo el concepto de mejor individuo. Aquí no podemos hacer uso de un procedimiento de ranking debido a que el tamaño de esta población es generalmente pequeño (típicamente, el $RefSet$ está formado en total por unos 20 individuos o menos). Nuestro enfoque es comparar cada nueva solución i con todos los individuos en $RefSet_1$ usando un test de dominancia. Así, cuando un individuo no es dominado por ninguna de las soluciones en $RefSet_1$, este se inserta en el conjunto de referencia sólo si este no está ya completo. Esto significa que el nuevo individuo debe dominar al menos una solución en $RefSet_1$ para poder insertarlo. Si esta condición no se cumple, el individuo se envía al archivo externo.

7.3. Otras propuestas

Además de los GAs de estado estacionario y búsquedas dispersas utilizados para resolver los tres problemas de telecomunicaciones, hemos desarrollado un algoritmo específico para cada problema particular atendiendo a sus características. En concreto, hemos utilizado una estrategia evolutiva para resolver el problema de la planificación de celdas, un algoritmo basado en colonias de hormigas para el problema de la asignación de frecuencias y una algoritmo genético celular para el problema de la difusión óptima en MANETs. Los algoritmos trabajan con la formulación natural de cada problema, así que tanto la estrategia evolutiva como el algoritmo genético son propuestas para optimización multiobjetivo, mientras que el ACO es un algoritmo monoobjetivo.

7.3.1. pPAES: estrategias evolutivas para resolver ACP

Uno de los principales problemas del problema de la planificación de celdas o ACP (ver Capítulo 4) es que, para instancias del mundo real como las que estamos abordando en esta tesis, los tiempos de cómputo son muy elevados incluso para algoritmos aproximados como las metaheurísticas. Buscando la máxima eficiencia temporal, el estudio sobre este problema comenzó con la estrategia más simple capaz de generar un frente de Pareto: (1+1) PAES o, simplemente, PAES [136]. Es una estrategia evolutiva en la que no existe autoadaptación (ver Capítulo 3) ni recombinación (sólo se utiliza el operador de mutación) que, además, incorpora un archivo externo para almacenar las soluciones no dominadas.

El pseudocódigo de PAES se muestra en el Algoritmo 7.3. Este algoritmo se basa en mantener una única solución que, en cada iteración, se muta para generar una nueva solución candidata (línea 5). Seguidamente, hay que determinar si la nueva solución generada pasa a ser la actual y si se inserta o no en el archivo externo. Estas decisiones se toman atendiendo a la dominancia de Pareto y a la distribución de soluciones en el frente (Algoritmo 7.4). PAES utiliza una medida de diversidad basada en un grid adaptativo para distribuir uniformemente las soluciones no dominadas en el frente resultante [136].

Algoritmo 7.3 Pseudocódigo de (1 + 1)-PAES.

```
1: archivo ← ∅
2: generar aleatoriamente la solución inicial c
3: insertar(archivo, m)
4: while no se cumple la condición de terminación do
5:    m ← mutar(c)
6:    evaluar(m)
7:    if c domina a m then
8:       descartar m
9:    else if m domina a c then
10:      insertar(archivo, m)
11:   else if m es dominada por cualquier miembro del archivo then
12:      descartar m
13:   else
14:      test(c, m, archivo)
15:   end if
16: end while
```

Con la intención, no sólo de reducir los tiempos de cómputo sino también de explorar mejor el espacio de búsqueda del ACP, hemos desarrollado un modelo paralelo de PAES, denominado pPAES, en el que múltiples islas colaboran entre sí. Cada una de estas islas ejecuta la versión de PAES secuencial, manteniendo un archivo local de soluciones no dominadas. Este archivo tiene el mismo tamaño máximo en cada isla, N, y, además de servir de repositorio, es una opción mucho más eficiente que tener un frente único que es actualizado por cada subalgoritmo ya que, en plataformas de cómputo distribuidas, esto significaría realizar comunicaciones remotas.

La colaboración entre las islas de pPAES se realiza mediante el intercambio periódico de soluciones no dominadas. La topología de comunicaciones utilizada en este proceso de migración es un

Algoritmo 7.4 Pseudocódigo de la función *test(c,m,archivo)*.

```
1: if el archivo no está lleno then
2:    insertar(archivo,m)
3:    if m está en una región menos poblada que c then
4:       c ← m //m pasa a ser la solución actual
5:    end if
6: else
7:    if m está en la región menos poblada del archivo then
8:       insertar(archivo,m)
9:       eliminar solución de la región más poblada
10:      if m está en una región menos poblada que c then
11:         c ← m //m pasa a ser la solución actual
12:      end if
13:   else
14:      if m está en una región menos poblada que c then
15:         c ← m //m pasa a ser la solución actual
16:      end if
17:   end if
18: end if
```

anillo unidireccional síncrono en el que una solución seleccionada aleatoriamente de cada archivo local se envía al vecino correspondiente en el anillo, que trata de incluirlo en su frente local siguiendo el esquema mostrado en la Figura 7.4. Estas migraciones se producen cada cierto número predeterminado de pasos de los algoritmos PAES locales.

El último paso de pPAES consiste en construir la aproximación final al frente de Pareto que se presenta como resultado del algoritmo. El objetivo es conseguir un frente con N soluciones no dominadas, es decir, el mismo tamaño que posee cada frente local. La decisión de dejar que cada isla explorase el espacio de búsqueda completo y que generase un frente de este tamaño tiene que ver con la imposibilidad de subdividir a priori el espacio de búsqueda de cualquier problema de optimización multiobjetivo para que se explore únicamente una parte del frente de Pareto. Así, si tenemos i islas, tendremos al final, al menos, $i \times N$ soluciones no dominadas (ya que no todas las islas tienen porqué llenar su frente). Una isla distinguida se encarga entonces de recolectar toda las soluciones de las demás y de generar el frente único utilizando el grid adaptativo propio de PAES.

7.3.2. Colonias de hormigas para la resolución del AFP

La propuesta algorítmica para resolver el problema de la asignación de frecuencias es un algoritmo basado en colonias de hormigas (ACO). Esta elección es debida a que este tipo de metaheurística se comporta muy bien con problemas que se pueden modelar como la búsqueda de caminos mínimos en grafos, como es el caso de la planificación de frecuencias. A pesar de ser un problema muy conocido y de adaptarse bien al problema, se pueden encontrar pocos trabajos en la literatura sobre ACOs para la resolución del diferentes versiones del AFP [3, 168, 184]. Dado que han reportado soluciones competitivas en muchos casos, consideramos que su inclusión en este estudio es de gran interés.

Los ACOs son un tipo de metaheurística basada en población cuya filosofía está inspirada en el comportamiento de las hormigas reales cuando buscan comida [29]. La idea principal consiste en usar hormigas artificiales que simulan dicho comportamiento en un escenario también artificial: un grafo. Las hormigas artificiales se colocan en nodos iniciales de dicho grafo y lo recorren saltando de un nodo a otro con la meta de encontrar el camino más corto desde su nodo inicial hasta un nodo final u objetivo. Cada hormiga avanza de forma independiente a las demás, pero la decisión del siguiente nodo a visitar depende de ciertos valores numéricos asociados a los arcos o nodos del grafo. Estos valores modelan los *rastros de feromona* que depositan las hormigas reales cuando caminan. Las hormigas artificiales alteran (al igual que las hormigas reales) los rastros de feromona del camino trazado, de modo que el avance de una puede influir en el camino de otra. De esta forma se incorpora a los ACO un mecanismo de cooperación indirecta entre las distintas hormigas simuladas, que constituye un factor clave en la búsqueda [72].

En un ACO las soluciones candidatas se representan mediante una secuencia de *componentes* escogidos de un conjunto de componentes C. De hecho, una cuestión importante cuando se resuelve un problema de optimización con un ACO es la elección del conjunto de componentes C y el modo en que las soluciones se construyen usando estos componentes. En algunos problemas esta cuestión es trivial porque las soluciones son ya una secuencia de elementos. Éste es el caso del problema del viajante de comercio (*Traveling Salesman Problem*, TSP), donde una solución es una secuencia de ciudades. Para otros problemas, como el entrenamiento de redes neuronales, la representación no es tan directa [228]. Las hormigas artificiales construyen la solución usando un procedimiento constructivo estocástico. Durante esta fase de construcción las hormigas caminan aleatoriamente por un grafo $G = (C, L)$ llamado *grafo de construcción*, donde L es un conjunto de *conexiones* (arcos) entre los componentes (nodos) de C. Cada arco $l_{ij} \in L$ tiene asociado un rastro de feromona τ_{ij} y puede tener asociado también un valor heurístico η_{ij}. Ambos valores se usan para guiar la fase de construcción estocástica que realizan las hormigas, pero hay una diferencia importante entre ellos: los rastros de feromona se modifican a lo largo de la búsqueda, mientras que los heurísticos son

Algoritmo 7.5 Pseudocódigo de ACO.

```
1: entrada: instancia del problema
2: p_bs ← NULL, p_rb ← NULL, cf ← 0, bs_update ← false
3: InicializarValoresFeromonas(T)
4: while no se cumpla la condición de terminación do
5:    for j ← 1 to n_a do
6:       p_j ← GenerarSolución(T)
7:    end for
8:    p_ib ← argmin(C(p_1), ..., C(p_{n_a}))
9:    p_ib ← BúsquedaLocal(p_ib,d)        /* opcional */
10:   Actualizar(p_ib,p_rb,p_bs)
11:   AplicarActualizaciónFeromonas(cf,bs_update,T,p_ib,p_rb,p_bs)
12:   cf ← CalcularFactorConvergencia(T)
13:   if cf > 0,99 then
14:      if bs_update = true then
15:         ReiniciarValoresFeromona(T)
16:         p_rb ← NULL
17:         bs_update ← false
18:      else
19:         bs_update ← true
20:      end if
21:   end if
22: end while
23: salida: p_bs
```

valores fijos establecidos por fuentes externas al algoritmo (el diseñador). Los rastros de feromona pueden asociarse también a los nodos del grafo de construcción (componentes de la solución) en lugar de a los arcos (conexiones entre componentes). Esta variación es especialmente adecuada para problemas en los que el orden de los componentes no es relevante (problemas de subconjunto [149]).

La propuesta particular que hemos utilizado en esta tesis se conoce como $\mathcal{MM}$AS en el *hypercube framework* (HCF) [28]. El Algoritmo 7.5 incluye una descripción de alto nivel de este algoritmo. Las estructuras de datos que utiliza, además de los contadores y del modelo de feromonas $\mathcal{T}$ son:

1. la mejor solución generada en la iteración actual, p_{ib},
2. la mejor solución generada desde el comienzo de la ejecución, p_{bs},
3. la mejor solución generada desde el último reinicio del algoritmo, p_{rb},
4. el factor de convergencia *cf*, $0 \leq cf \leq 1$, que mide cómo de lejos está el algoritmo de converger, y
5. la variable lógica *bs_ update*, que se hace cierta cuando el algoritmo converge.

El algoritmo funciona como sigue. En primer lugar, se inicializan todas las variables. En particular, el procedimiento InicializarValoresFeromonas($\mathcal{T}$) establece los valores de feromona a 0,5. Cada iteración del algoritmo consiste en los siguientes pasos. Primero, se generan n_a soluciones utilizando la función GenerarSolución($\mathcal{T}$). La mejor solución de la iteración p_{ib} puede ser mejorada por la aplicación opcional de un método de búsqueda local. Segundo, la función Update(p_{ib},p_{rb},p_{bs} actualiza los valores de las variables p_{ib}, p_{rb} y p_{bs}. Tercero, se actualizan los rastros de feromona mediante el uso de la función AplicarActualizaciónFeromonas(*cf*, *bs_ update*, $\mathcal{T}$, p_{ib}, p_{rb}, p_{bs}). Cuarto, se calcula

un nuevo valor para el factor de convergencia *cf*. Dependiendo de este valor, así como de la variable lógica *bs_ update*, se decide si reiniciar o no el algoritmo. Si se reinicia, se invoca el procedimiento ReiniciarValoresFeromona($\mathcal{T}$) y todos los valores de las feromonas se establecen a 0,5. El algoritmo itera hasta que se satisface alguna condición de terminación.

Tanto la generación de las soluciones como la búsqueda local, es decir, los métodos GenerarSolución y BúsquedaLocal, son dependientes del problema y, por tanto, su descripción se deja para el Capítulo 9 junto con los métodos para ssGA y SS. Incluimos aquí, no obstante, cómo se realiza la actualización de la matriz de feromonas (InicializarValoresFeromonas) y cómo se calcula el factor de convergencia (CalcularFactorConvergencia).

AplicarActualizaciónFeromonas(cf,$bs\_\ update$,$\mathcal{T}$,p_{ib},p_{rb},p_{bs})

En general, se utilizan tres soluciones para actualizar los valores de las feromonas: la mejor solución de la iteración p_{ib}, la mejor solución desde el último reinicio p_{rb} y la mejor solución encontrada hasta el momento desde el comienzo de la ejecución p_{bs}. La influencia de cada solución en el proceso de actualización de feromonas depende del estado de convergencia del algoritmo que se mide con el factor de convergencia *cf*. Al comienzo de cada reinicio (por ejemplo, cuando *bs_ update* = FALSE y *cf* es próximo a cero), la mejor solución de la iteración tiene influencia máxima. Entonces, a la vez que el algoritmo progresa (cuando *cf* se incrementa, por ejemplo), la influencia de p_{rb} crece. Justo antes de converger ($cf \leq 0{,}99$, pero con valor cercano a 0,99), es la solución p_{rb} la que tiene máxima influencia, y, una vez que se ha detectado la convergencia del algoritmo ($cf > 0{,}99$), la variable de control *bs_ update* pasa a ser TRUE, lo que provoca que p_{bs} sea la que tenga mayor efecto. Esto se hace para intensificar la búsqueda en la zona de la mejor solución encontrada y puede tener el efecto de dirigir la exploración hacia otras zonas del espacio de búsqueda, con el consecuente decremento del factor de convergencia. Así, cada valor de feromona $\tau_{ij} \in \mathcal{T}$ se actualiza como sigue:

$$\tau_{ij} \leftarrow \tau_{ij} + \rho \cdot (\mu_{ij} - \tau_{ij}) \ , \tag{7.1}$$

donde

$$\mu_{ij} \leftarrow \kappa_{ib} \cdot \delta(p_{ib}, f_{ij}) + \kappa_{rb} \cdot \delta(p_{rb}, f_{ij}) + \kappa_{bs} \cdot \delta(p_{bs}, f_{ij}) \ , \tag{7.2}$$

donde κ_{ib} es el peso (es decir, la influencia) de la solución p_{ib}, κ_{rb} es el peso de la solución p_{rb}, κ_{bs} es el peso de la solución p_{bs}, y $\kappa_{ib} + \kappa_{rb} + \kappa_{bs} = 1$. Además, $\delta(p, f_{ij}) = 1$ si $p(t_i) = f_{ij}$, y $\delta(p, f_{ij}) = 0$, en otro caso. Después de aplicar la regla de actualización de feromonas (Ecuación 7.1), a los valores que exceden $\tau_{\text{máx}} = 0{,}999$ se les fija su valor a $\tau_{\text{máx}}$ (igualmente para $\tau_{\text{mín}} = 0{,}001$).

La Ecuación 7.2 permite elegir cómo se planifica la influencia relativa de las tres soluciones utilizadas para actualizar los valores de feromonas. En este caso concreto, la Tabla 7.1 muestra la planificación que se ha utilizado.

Tabla 7.1: Valores de κ_{ib}, κ_{rb}, κ_{bs} y ρ dependiendo del factor de convergencia *cf* y de la variable de control *bs_ update*.

	bs_ update = FALSE				*bs_ update*
	$cf < 0{,}4$	$cf \in [0{,}4, 0{,}6)$	$cf \in [0{,}6, 0{,}8)$	$cf \geq 0{,}8$	= TRUE
κ_{ib}	1	2/3	1/3	0	0
κ_{rb}	0	1/3	2/3	1	0
κ_{bs}	0	0	0	0	1
ρ	0,2	0,2	0,2	0,15	0,15

CalcularFactorConvergencia($\mathcal{T}$)

El factor de convergencia cf, cuyo valor está basado en los valores actuales de feromona, se calcula como sigue:

$$cf \leftarrow 2 \left(\left(\frac{\sum_{\tau_{ij} \in \mathcal{T}} \text{máx}\{\tau_{\text{máx}} - \tau_{ij}, \tau_{ij} - \tau_{\text{mín}}\}}{|\mathcal{T}| \cdot (\tau_{\text{máx}} - \tau_{\text{mín}})} \right) - 0{,}5 \right) \ .$$

De esta forma, $cf = 0$ cuando el algoritmo se inicia (o reinicia), esto es, cuando todos los valores de feromona son 0,5. Por otro lado, cuando el algoritmo ha convergido, $cf = 1$. En el resto de casos, cf tiene un valor en $(0, 1)$.

7.3.3. Algoritmos genéticos celulares para MANETs: cMOGA y MOCell

La utilización de algoritmos genéticos para la resolución de este novedoso problema tiene que ver con la habilidad de éstos para explorar espacios de búsqueda complejos sobre los que se tiene poco conocimiento, y ése es precisamente nuestro caso al ser un problema en el que la propia función de valuación implica simulaciones estocásticas. El uso del modelo celular, además de estar demostrado en la literatura que ofrece mejores soluciones que el GA panmíctico en numerosos problemas de optimización, incluye un componente histórico fundamental que tiene que ver con la experiencia de los autores dentro de este campo y con que, después de revisar la literatura, nos encontramos que no existía algoritmo genético celular multiobjetivo alguno que siguiera el modelo canónico, tal y como se describe en la literatura [75]. Así, en [9, 11] presentamos tanto el nuevo problema de optimización como cMOGA (*cellular MultiObjective Genetic Algorithm*), el primer cGA canónico para optimización multiobjetivo.

En el Algoritmo 7.6, damos el pseudocódigo de cMOGA. Se trata de un algoritmo celular canónico [75] que sigue el modelo síncrono al que se le incorpora un archivo externo para almacenar soluciones no dominadas (línea 2). Así, los individuos se sitúan en una rejilla toroidal de 2 dimensiones y se le aplican sucesivamente los operadores genéticos (líneas 7 y 8) hasta que llegamos a la condición de parada (línea 3). Para cada individuo el algoritmo selecciona dos padres de su vecindario, los recombina para obtener un descendiente, que es mutado. Después se evalúa al individuo resultante, que se inserta tanto en la población auxiliar como en el frente de Pareto (utilizando la

Algoritmo 7.6 Pseudocódigo de cMOGA.

```
1: proc Evoluciona(cmoga)        //Parámetros del algoritmo en 'cmoga'
2: Frente_Pareto = CrearFrente() //Crea un Frente de Pareto vacío
3: while !CondiciónParada() do
4:    for individuo ← 1 hasta cmoga.TamañoPobl do
5:       vecindario ← CalculaVecindario(cmoga,posición(individuo));
6:       padres ← Selección(vecindario);
7:       descendiente ← Recombinación(cmoga.Pc,padres);
8:       descendiente ← Mutación(cmoga.Pm,descendiente);
9:       Evaluación(descendiente);
10:      Reemplazo(posición(individuo),pobl_auxiliar,descendiente);
11:      InsertarFrentePareto(individuo);
12:   end for
13:   cmoga.pop ← pobl_auxiliar;
14: end while
```

Algoritmo 7.7 Pseudocódigo de MOCell.

```
1: proc Evoluciona(mocell)          //Parámetros del algoritmo en 'mocell'
2: Frente_pareto = Crear_Frente() //Crea un frente de Pareto vacío
3: while !CondiciónParada() do
4:   for individuo ← 1 hasta mocell.Tamaño_Población do
5:     vecinos ← ObtenerVecindario(mocell,posición(individuo));
6:     padres ← Selección(vecinos, Frente_pareto);
7:     descendiente ← Recombinación(mocell.Pc,padres);
8:     descendiente ← Mutación(mocell.Pm,descendiente);
9:     Evaluar(descendiente);
10:    Insertar(vecinos,descendiente,mocell);
11:    InsertarFrentePareto(individuo, Frente_pareto);
12:  end for
13: end while
```

rejilla adaptativa de PAES [136]) si el individuo actual no lo domina –líneas 10 a 13. Finalmente, en cada generación, la población auxiliar reemplaza a la población antigua (modelo síncrono).

Sucesivos trabajos posteriores [189, 190, 191] han ido mejorando cMOGA hasta llegar a una nueva propuesta algorítmica basada en el modelo celular a la que hemos denominada MOCell que, junto con AbYSS, son unos de los algoritmos más eficientes y efectivos para diferentes bancos de prueba del dominio. Queremos indicar aquí que la versión de MOCell que se utiliza en esta tesis es la denominada aMOCell4 en [191]. En ese trabajo se analizaron diferentes configuraciones con estrategias síncronas/asíncronas, así como diversas formas de realimentación de soluciones del frente de Pareto encontrado, y aMOCell4 es la que proporcionó los resultados más precisos. Para simplificar la notación, a partir de ahora utilizaremos MOCell para referirnos a aMOCell4. El pseudocódigo de este algoritmo puede verse en el Algoritmo 7.7. La primera diferencia con cMOGA es que MOCell sigue el modelo asíncrono (no existe población auxiliar). No obstante, al igual que AbYSS o cMOGA, incluye un archivo externo o frente de Pareto (línea 2). Para poder manejar la inserción de soluciones en el frente de Pareto con el objeto de obtener un conjunto diverso hemos utilizado un estimador de densidad basado en la distancia de *crowding* (propuesto en NSGA-II [68]). Este método también se usa para eliminar las soluciones del archivo cuando se llena. Ésta es otra diferencia con cMOGA, que utiliza la rejilla adaptativa de PAES.

MOCell empieza creando un frente de Pareto vacío (línea 2 del Algoritmo 7.7). Los individuos se sitúan en una malla toroidal de 2 dimensiones, y se les va aplicando sucesivamente el ciclo reproductor (líneas 4 a 12) hasta que alcanzamos la condición de parada (línea 3). Así, para cada individuo, el algoritmo selecciona dos padres por torneo binario, uno del vecindario y otro del archivo externo. He aquí la segunda diferencia fundamental con cMOGA, ya que en en el mecanismo de selección también interviene el frente almacenado (línea 6). La medida utilizada para elegir al ganador del torneo binario se corresponde con su distancia de crowding dentro del vecindario y del archivo, respectivamente. La selección de un padre del archivo introduce implícitamente la realimentación de soluciones del frente (intensificación) que va a permitir guiar la búsqueda hacia regiones prometedoras. Una vez se han seleccionado los padres, se recombinan para obtener un descendiente, que es posteriormente mutado y evaluado. Ahora hay que insertar el nuevo individuo tanto en la población como en el archivo externo. En el primer caso, este nuevo individuo reemplaza a la solución del vecindario de la solución actual con peor distancia de crowding. En el segundo caso y siguiendo el mismo esquema que AbYSS, la inserción en el frente usa el esquema de PAES, pero utilizando la distancia de crowding de NSGA-II como medida de diversidad en lugar de usar el grid adaptativo de cMOGA.

Para tratar con problemas con restricciones, MOCell utiliza la misma aproximación que NSGA-II para comparar dos soluciones. Si las dos son factibles, se les aplica directamente un análisis de dominancia de Pareto (Definición 8). Si una es factible y la otra no, el primero domina. En otro caso, si ninguno de los dos son factibles, el que viole las restricciones normalizadas (ver Sección 3.3) en menor cantidad, dominará al otro.

7.4. Extensiones de las propuestas para ejecución en sistemas de computación grid

Los sistemas de computación grid son capaces de ofrecer una enorme capacidad de cómputo que permite abordar problemas problemas que, de otra forma, sería imposible resolver en un tiempo razonable. No obstante, las características propias de este tipo de sistemas hacen que sea difícil la utilización directa de los modelos paralelos para metaheurísticas presentados en la Sección 3.4. El principal inconveniente es la dinamicidad de los recursos computacionales, es decir, el hecho de que en cualquier momento un recurso pueda aparecer o desaparecer del sistema. Esto, junto con el número potencialmente elevado de máquinas que pueden participar, impide la distribución de la computación utilizando topologías regulares (anillos, rejillas, etc.). Así, el modelo paralelo más apropiado para su despliegue en un sistema grid es el esquema maestro/esclavo, en el que el cálculo de la función de fitness se realiza de forma remota. En el contexto en el que nos movemos en esta tesis, resolviendo problemas de optimización de corte real e instancias de gran tamaño, el cómputo del fitness supone casi la totalidad de la carga computacional de los algoritmos.

Una implementación adecuada de un esquema centralizado basado en el modelo maestro/esclavo requiere afrontar, además, los siguientes problemas (ver Sección 3.4.3):

- Fallos en la máquina que ejecuta al proceso maestro.
- Fallos en la máquina que ejecuta a un esclavo .
- El algoritmo debe tener en cuenta la incorporación dinámica de recursos al sistema grid para usarlos tan pronto como sea posible.
- Diferentes tiempos de respuesta de los esclavos ya que las máquinas que componen el sistema suelen tener distinta potencia de cómputo o bien pueden existir retrasos en la red.
- Ajustar el grano de computación en los esclavos para evitar que el maestro sea un cuello de botella .

Todos estos aspectos se pueden abordar a diferentes niveles: software del sistema grid, metaheurística grid, y problema que se resuelve. Las diferentes opciones se muestran en la Tabla 7.2. Una opción entre paréntesis indica que es secundaria. Por ejemplo, un fallo en la máquina que ejecuta

Tabla 7.2: Aspectos a considerar en una metaheurística grid basada en el modelo maestro esclavo y los niveles de software en los que se debe abordar.

	software grid	**metaheurística grid**	**problema**
Fallos en el maestro	Sí (No)	Sí (Yes)	No
Fallos en un esclavo	No (Sí)	Sí (No)	No
Detección de nuevos recursos	Sí	Sí	No
Diferentes tiempos de respuesta	No	Sí	No
Grano de computación	No	No	Sí

al maestro se puede controlar a nivel de software del sistema grid si éste proporciona puntos de control automáticos (*automatic checkpointing*); en otro caso, debe ser la metaheurística la que maneje este aspecto. El mismo argumento se mantiene cuando falla un esclavo, aunque en este caso la respuesta de la metaheurística puede variar desde el reenvío de la solución a otro esclavo para su evaluación remota, hasta simplemente ignorar este fallo. Si se incorporan nuevos procesadores al sistema, es el software grid el encargado de proporcionar los mecanismos necesarios para notificarlo al algoritmo grid, que debe reaccionar consecuentemente. El hecho de que el tiempo de respuesta de los esclavos es variable involucra exclusivamente a la metaheurística grid. Finalmente, el ajuste del grano de computación de los esclavos depende directamente de la complejidad de la función de evaluación del problema de optimización correspondiente.

7.4.1. GrEA

El algoritmo GrEA [193, 164] es un GA con selección por estado estacionario que sigue el modelo maestro/esclavo. Se ha desarrollado sobre el sistema de computación grid Condor [238] y la biblioteca MW [154]. La idea básica es que un proceso maestro ejecuta el bucle principal del algoritmo ssGA y los esclavos realizan la evaluación de la función, de forma asíncrona. Así, al contrario de lo que ocurren en el ssGA secuencial, GrEA lanza varias evaluaciones en paralelo; idealmente habrá tantas evaluaciones paralelas como recursos computacionales haya en el sistema grid.

Para describir mejor el algoritmo, sean GrEA-master y GrEA-esclavo las partes del mismo que se corresponden con el proceso maestro y con los procesos esclavos, respectivamente. El pseudocódigo de GrEA-maestro se incluye en el Algoritmo 7.8. Éste comienza creando una población vacía (línea 1) y generando una lista de tareas, de forma que cada una de estas tareas contiene un individuo generado aleatoriamente (línea 2). Una vez creada, Condor envía las tareas de la lista a los esclavos disponibles para la evaluación de estos individuos. Después de estos dos pasos, GrEA-maestro trabaja de forma reactiva: cuando se recibe una tarea de un esclavo (línea 4), se extrae el individuo que contiene (línea 5) y se inserta en la población (línea 6). Entonces, el GrEA-maestro pregunta al sistema por los esclavos que están disponibles para computar y genera tantas tareas como esclavos como sigue: primero, se realiza una iteración del ssGA, es decir, selección, recombinación y mutación (línea 8); segundo, el individuo se añade a una nueva tarea (línea 9); finalmente, la tarea se inserta en la lista de tareas que el sistema Condor envía a los esclavos para su computación. Como se puede observar, el asincronismo es total, ya que las tareas se van creando en función del número de esclavos disponibles y se van insertando a en el orden de llegada, no en el envío, por lo que se pueden incorporar a la población individuos que se generaron antes, pero

Algoritmo 7.8 Pseudocódigo de GrEA-master.

```
1:  poblacion ← ∅
2:  Inicializar listaDeTareas
3:  while no se cumpla la condición de terminación do
4:    Recibir tarea
5:    individuo ← tarea.individuo
6:    Insertar individuo en poblacion
7:    while no hay esclavos disponibles do
8:      nuevoIndividuo ← GA_step()
9:      nuevaTarea ←new Tarea(nuevoIndividuo)
10:     listaDeTareas.insertar(nuevaTarea)
11:   end while
12: end while
```

Algoritmo 7.9 Pseudocódigo de GrEA-worker.

```
1: while true do
2:    Recibir tarea
3:    individuo ← tarea.individuo
4:    nuevoIndividuo ← BúsquedaLocal(individuo)
5:    nuevaTarea ←new Tarea(nuevoIndividuo)
6:    Return nuevaTarea
7: end while
```

cuya ejecución remota se ha realizado en una máquina más potente.

La misión de los GrEA-esclavos consiste en recibir un individuo, evaluarlo y, opcionalmente, aplicarle algún método de búsqueda local que permita ajustar el tiempo de cómputo de los esclavos y así hacer que el ratio comunicación/computación sea favorable. Dado que la búsqueda local generalmente modifica el individuo, éste debe ser enviado de nuevo al GrEA-master. El pseudocódigo del GrEA-esclavo se puede observar en el Algoritmo 7.9.

Algunas de las características más importantes de GrEA es la detección de nuevos procesadores durante la computación y la tolerancia a fallos. Estas características juegan un papel fundamental para hacer GrEA un algoritmo que se puede ejecutar en sistemas de computación grid. Así, cada vez que GrEA detecta un nuevo procesador disponible, se crea un nuevo GrEA-esclavo y se realiza un paso del GA, que genera un individuo para ser evaluado en el nuevo esclavo creado. Respecto a la tolerancia a fallos, las caídas del GrEA-maestro las maneja Condor automáticamente utilizando puntos de control (*checkpoints*); los fallos en los esclavos simplemente se ignoran ya que no afectan al funcionamiento del GA.

7.4.2. assNSGA-II

El algoritmo assNSGA-II es la versión grid de ssNSGA-II, el algoritmo NSGA-II con selección por estado estacionario propuesto en esta tesis. La idea para distribuir en un sistema de computación assNSGA-II es la misma que GrEA: un esquema asíncrono maestro-esclavo. Las únicas diferencias tienen que ver con el hecho de que esta nueva propuesta es para problemas multiobjetivo y, por tanto, para los métodos de selección y el reemplazo no se utiliza el fitness, sino el ranking y crowding propios de NSGA-II. La otra diferencia está en la implementación. Si bien GrEA está implementado sobre Condor/MW, aquí hemos utilizado *Sparrow*, una herramienta java basada en MW para implementar aplicaciones maestro/esclavo en grids, que se está desarrollando actualmente en nuestro grupo de investigación.

7.5. Conclusiones

En este capítulo hemos presentado las propuestas metodológicas utilizadas para resolver los tres problemas de redes de telecomunicaciones con los que estamos trabajando en esta tesis. Para dar coherencia al trabajo, hemos utilizado dos metaheurísticas para resolver los tres problemas: algoritmos evolutivos y búsqueda dispersa, de las que se han desarrollado versiones tanto mono como multiobjetivo. Además, para cada uno de los problemas, hemos presentado también un algoritmo que se adecua específicamente a sus características. El capítulo termina presentando las propuestas para la extensión de algunos de los algoritmos diseñados para que se puedan ejecutar de forma eficiente en sistemas de computación grid.

Capítulo 8

Resolución del problema ACP

Este capítulo aborda la resolución del problema ACP con los tres algoritmos presentados en el Capítulo 7: ssNSGA-II, AbYSS y pPAES. Hemos utilizado las versiones multiobjetivo de las propuestas debido a la naturaleza propia del problema que tiene tres funciones que deben ser optimizadas a la vez: minimizar el número de antenas instaladas, minimizar las interferencias dentro de la red y maximizar el tráfico de red, todo sujeto a dos restricciones de cobertura y *handover*.

La primera sección que incluimos en este capítulo (Sección 8.1) se dedica a una revisión de la literatura sobre los trabajos relacionados con la utilización de algoritmos evolutivos y búsqueda dispersa para la resolución de este problema. Dado que ssNSGA-II, AbYSS y pPAES se presentaron ya en el capítulo anterior, sólo queda definir la representación y los operadores concretos que utilizan. Esto se hace en la Sección 8.2. En la Sección 8.3 hemos incluido las tres instancias reales del problema que han sido abordadas. La Sección 8.4 está dedicada a presentar los experimentos realizados y al análisis de resultados, comparando nuestras propuestas con los dos algoritmos del estado del arte en optimización multiobjetivo, NSGA-II [68] y SPEA2 [262]. El capítulo acaba con las principales conclusiones alcanzadas.

8.1. Trabajos relacionados

En esta sección se incluye una revisión de la literatura para mostrar el uso tanto de algoritmos evolutivos en general, y genéticos en particular, como de la búsqueda dispersa en la resolución del problema de la planificación de celdas. Si bien el número de trabajos en el primer caso es muy elevado, nunca se ha utilizado búsqueda dispersa, hasta donde conocemos, para resolver este problema.

8.1.1. Algoritmos Evolutivos

Los algoritmos evolutivos, y más concretamente los algoritmos genéticos, han sido utilizados ampliamente en la literatura para resolver la mayoría de las formulaciones del ACP. En esta sección, hemos intentado incluir el mayor número de trabajos posibles que reflejen al amplio abanico de EAs utilizados para las distintas aproximaciones al problema. Han sido revisados más de 40 trabajos, cuyo resumen se incluye en las Tablas 8.1, 8.2 y 8.3. Por cada fila, las tablas muestran la siguiente información:

- Ref.: Referencias bibliográficas donde se presenta el trabajo.
- Año: El año de la primera publicación relacionada con el trabajo concreto.
- Alg.: Incluye el tipo de algoritmo concreto que se utiliza en la resolución del problema. Aquí distinguimos entre:
  - GA: Algoritmo genético simple. Utilizamos este nombre cuando no se da información suficiente en la publicación para poder especificar más.
  - genGA: Algoritmo genético generacional.
  - ssGA: Algoritmo genético de estado estacionario.
  - dGA: Algoritmo distribuido siguiendo el modelo de islas. Puede aparecer combinado con los dos anteriores. Así, dssGA hace referencia a un modelo distribuido que en cada isla utiliza un GA de estado estacionario.
  - ES: Estrategia evolutiva.
  - Algoritmos concretos como CHC [81], DE (Evolución Diferencial) [233], NSGA-II [68], SPEA2 [262], y otros menos conocidos, cuya descripción se puede encontrar revisando la referencia correspondiente que se proporciona.
- Multi: Este campo indica si se resuelve el problema utilizando técnicas específicas para optimización multiobjetivo basadas en optimalidad de Pareto.
- Modelo ACP: Categoría del modelo ACP utilizado para resolver el problema, según se explicó en el Capítulo 4. Los posibles valores son Nodos de demanda, Discos y Puntos de test (ver Sección 2.1.1).
- Emplazamientos: Muestra si el emplazamiento de las BTSs se elige de entre un conjunto de lugares potenciales (CSL o *Candidate Site List*) o bien se pueden posicionar libremente en cualquier punto de la zona geográfica.
- Celda: Este campo hace referencia a cómo se calcula la celda o área de servicio de cada BTS.
- Pw, Ti y Az: Indican si se optimizan los parámetros de potencia de emisión y los ángulos de inclinación y acimut de las BTSs.
- Objetivos: Esta columna indica los diferentes aspectos de la red que se optimizan.
- Restricciones: Incluye los aspectos de la red que se consideran como restricciones en el proceso de optimización. La columna incluye el símbolo − si el no existe restricción alguna.

Desde el punto de vista algorítmico, se han utilizado GAs clásicos generacionales (genGA) y aquellos que utilizan selección de estado estacionario (ssGA), así como algoritmos muy particulares como CHC [81], Evolución Diferencial [233] o PBIL [23]. Desde el punto de vista del paralelismo, si bien se pueden encontrar en la literatura modelos distribuidos en islas (dGAs) [10, 37], no hemos encontrado ningún trabajo que aplique el modelo celular para abordar el ACP. Si revisamos los algoritmos evolutivos multiobjetivo, vemos que los métodos de referencia en el campo como NSGA-II y SPEA2 han sido aplicados, junto con alguna propuesta específica como SEAMO [207] o MOCHC [188]. Desde el punto de vista de la formulación, si bien en las primeras aproximaciones al problema se seguía un enfoque monoobjetivo en el que se ponderaban los distintos aspectos de la red que se querían optimizar (función agregativa) [36, 41, 150, 218], el auge en los últimos años de los algoritmos evolutivos para optimización multiobjetivo ha generado un buen número de trabajos que utilizan esta formulación [40, 188, 205, 236].

Tabla 8.1: EAs en la literatura para la resolución del problema ACP (I).

Ref.	Año	Alg.	Multi	Modelo ACP	Sites	Espacio de búsqueda Celda	*Pw*	*Ti*	*Az*	Objetivos	Restricciones
[35, 36, 37]	1997	dGA	•	Nodos de demanda	CSL	Modelo de propagación	•	•	•	Coste, cobertura	–
[41]	1998	dssGA	•	Nodos de demanda	CSL	Modelo de propagación	•	•	•	Coste, cobertura	–
[150]	1998	genGA	•	Nodos de demanda	CSL	Sintéticas	•	•	•	Cobertura	4 BTSs
[218]	1999	genGA	•	Puntos de test	CSL	Modelo de propagación	•	•	•	Cobertura, coste (dinámicos)	–
[181]	1999	genGA	•	Nodos de demanda	CSL	Modelo de propagación	•	•	•	Cobertura, coste	–
[34, 177, 236, 237]	2000	ssGA, híbrido	✓	Puntos de test	CSL	Modelo de propagación	✓	✓	✓	Coste, tráfico, interferencias	Cobertura, handover
[117]	2000	GA	•	Discos	Libre	Omnidireccional	✓	•	✓	Cobertura, tráfico, coste, interferencias	–
[259, 261, 260]	2000	ES	•	Puntos de test	CSL	Modelo de propagación	✓	✓	✓	Coste, interferencia, forma de la celda	Tráfico, cobertura
[147]	2000	dGA	•	Nodos de demanda	CSL	Modelo de propagación	•	•	•	Coste de expansión	Tráfico, capacidad, cobertura
[106]	2001	ssGA	•	Discos	Libre	Omnidireccional	•	•	•	Coste, Cobertura	–

Continúa...

Tabla 8.2: EAs en la literatura para la resolución del problema ACP (II).

Ref.	Año	Alg.	Multi	Modelo ACP	Sites	Espacio de búsqueda Celda	Pw	Ti	Az	Objetivos	Restricciones
[144]	2001	genGA	•	Puntos de test	Libre	Trazado de rayos	•	•	•	Cobertura, Retraso	–
[16, 17]	2002	GA	•	Puntos de test	CSL	Modelo de propagación	✓	✓	✓	Coste, cobertura, capacidad	Handover
[198]	2002	ssGA	•	Discos	Libre	Modelo de propagación	✓	•	•	Coste, Cobertura	–
[38, 39, 40]	2003	GA binario	•	Nodos de demanda	Libre	Omnidireccional	✓	✓	•	Radiación, tráfico, interferencia, eficiencia, cobertura	–
[250]	2003	SPEA2, NSGA-II, stEAPT	✓	Nodos de demanda	Libre	Omnidireccional	✓	•	•	Coste, Interferencias	Cobertura
[10, 7]	2004	ssGA, dssGA	•	Nodos de demanda	CSL	Cuadrada	•	•	•	Coste, Cobertura	–
[33, 42]	2004	GA, híbrido GA-TS	•	Puntos de test	CSL	Modelo de propagación	✓	•	•	Diferentes condiciones de cobertura	Handover, capacidad
[126, 127, 202]	2004	NSGA-II	✓	Puntos de test	CSL	Modelo de propagación	✓	✓	✓	Coste, cobertura, capacidad	Handover
[129]	2004	MOGA	✓	Puntos de test	CSL	Modelo de propagación	✓	✓	✓	Solapamiento, geometría	Cobertura
[153]	2004	GA binario	•	Nodos de demanda	CSL	Modelo de propagación	✓	•	•	Coste, cobertura, tráfico, handover	–

Continúa...

Tabla 8.3: EAs en la literatura para la resolución del problema ACP (y III).

Ref.	Año	Alg.	Multi	Modelo ACP	Sites	Espacio de búsqueda Celda	Pw	Ti	Az	Objetivos	Restricciones
[169]	2004	dGA	✓	Discos	CSL	Modelo de propagación	✓	•	•	Capacidad, cobertura, coste	–
[208, 207, 253, 254]	2004	SEAMO, SPEA2, NSGA-II, PESA	✓	Puntos de test	CSL	Modelo de propagación	✓	•	•	Coste, Cobertura	Handover
[258]	2004	MOGA, EMOGA	✓	Discos	Libre	Omnidireccional	•	•	•	Coste, Cobertura	–
[54]	2005	dES híbrida	•	Nodos de demanda	CSL	Hexagonal	•	•	•	Tráfico, geometría, coste, solapamiento	–
[13]	2007	CHC, ssGA, genGA	•	Nodos de demanda	CSL	Cuadrada, Omnidireccional, Directiva	•	•	✓	Coste, Cobertura	–
[188]	2007	MOCHC, NSGA-II	✓	Nodos de demanda	CSL	Cuadrada	•	•	•	Coste, Cobertura	Coste máximo, cobertura mínima
[205]	2007	NSGA-II	✓	Puntos de test	CSL	Modelo de propagación	•	•	•	Coste, Cobertura	Tráfico, handover
[246, 247]	2007	CHC, PBIL, DE	•	Nodos de demanda	CSL	Cuadrada	•	•	•	Coste, Cobertura	–

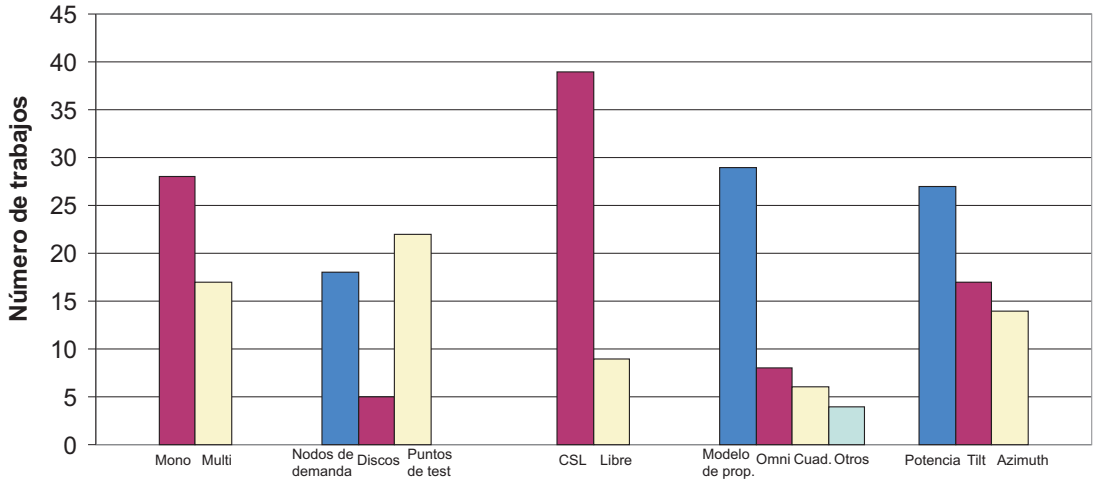


Figura 8.1: Resumen de trabajos relacionados con la resolución del ACP utilizando EAs.

A modo de resumen, la Figura 8.1 muestra el número de trabajos revisados que caen en diferentes categorías: mono/multi, modelo ACP utilizado, selección de emplazamientos, cálculo de la celda y optimización de parámetros de las BTSs. Discutamos ahora brevemente cada grupo de columnas que aparece en la figura. En primer lugar, de entre los trabajos revisados, la utilización de formulaciones monoobjetivo para resolver el ACP con GAs ha tenido una mayor incidencia, a pesar de que la formulación multiobjetivo es más "natural" por las características del problema. La complejidad adicional de los algoritmos multiobjetivo introducida por los conceptos de optimalidad de Pareto hace que los investigadores en el campo sean un poco reacios a adoptar este tipo de técnicas. No obstante, creemos, y de hecho es el enfoque adoptado en esta tesis, que la aproximación multiobjetivo es la más adecuada ya que va a permitir al experto disponer de un conjunto de planificaciones de celdas para su red, ninguna mejor que las demás, que podrá utilizar en diferentes escenarios atendiendo a las necesidades de la situación concreta en la que se encuentre.

De los tres modelos existentes para abordar el problema, sin duda el modelo de nodos de demanda, por su simplicidad y baja carga computacional, y el de puntos de test, debido a su ajuste con el mundo real, son los más utilizados. El modelo de discos tiene que ver más con estudios teóricos ya que, en la actualidad, rara vez nos encontramos con redes compuestas únicamente por antenas omnidireccionales (la sectorización permite aumentar la capacidad de la red considerablemente). Si nos fijamos en el tercer grupo de columnas de la Figura 8.1, se puede observar que la utilización de un conjunto de emplazamientos candidatos (CSL) en lugar de poder posicionar libremente las BTSs en cualquier parte de la red es claramente mayoritaria. Esto tiene que ver con las limitaciones que encuentran los operadores de telefonía móvil en la realidad cuando quieren desplegar una red: no se puede colocar una BTS en medio de una autovía, ni sobre un colegio, etc. El cuarto grupo de columnas de la figura también pone de manifiesto una opción preferente a la hora de calcular la zona de servicio de las BTSs (celdas), la utilización de modelos de propagación como el de espacio libre (*free-space*), Okumura-Hata o Walfish-Ikegami [50]. La elección de uno u otro depende normalmente de la carga computacional que suponga [123]. Las celdas omnidireccionales y cuadradas como modelo abstracto también aparecen en algunos trabajos (8 y 6 contribuciones, respectivamente, de entre las revisadas). Otros métodos que aparecen en las Tablas 8.1, 8.2 y 8.3 utilizan modernas técnicas de trazado de rayos (*ray tracing* [144]) o han sido generadas sintéticamente [150], entre otras. Finalmente, el último grupo de columnas muestra el número de trabajos en los que los

parámetros de potencia y ángulos de inclinación y acimut de las BTSs entran dentro del proceso de optimización. Es decir, son variables de decisión del espacio de búsqueda. Aunque aquí las diferencias son menores, se puede observar que la potencia es el valor que se optimiza con más frecuencia, ya que se puede aplicar a todo tipo de BTS (omnidireccional, directiva, cuadrada, etc.) como parámetro principal para controlar el tamaño de la misma. Los ángulos de inclinación y acimut se suelen utilizar para modelos ACP muy precisos y suponen una carga computacional considerable, de ahí su menor incidencia en los trabajos revisados.

Para concluir con la discusión sobre los trabajos relacionados con la resolución del ACP con algoritmos genéticos, nos fijamos ahora en las funciones objetivo y las restricciones utilizadas en las diferentes aproximaciones. Por parte de los objetivos, existe una tendencia clara que consiste en considerar coste de la red, medido en número de emplazamientos activados, y calidad del servicio que proporcionan estas BTSs. Ambos son claramente contradictorios. La diferencia entre muchos trabajos radica en lo que cada autor considera calidad del servicio. Maximizar cobertura o área de servicio de la red es lo más usual y aparece en el 78 % de los trabajos revisados. No obstante, un aproximación más realista se basa en considerarla como restricción (por ejemplo, un mínimo del 90 % del área ha de estar cubierta) para descartar aquellas soluciones que no son viables (no tiene sentido instalar una red que cubra un porcentaje pequeño del área de servicio). Éste es el enfoque seguido en esta tesis. Otras formas de medir la calidad de la red tienen en cuenta las interferencias que se provocan o el tráfico que es capaz de soportar. En cuanto a las restricciones, la que más aparece es el *handover*. La red debe garantizar la continuidad de la comunicación mientras el usuario se mueve entre diferentes celdas.

A pesar del gran número de trabajos, las contribuciones de esta tesis en este contexto son relevantes: es la primera vez que se aborda este problema con dos nuevos EAs, ssNSGA-II y pPAES, superando a los algoritmos del estado del arte NSGA-II y SPEA2. Además, hemos realizado un estudio sistemático sobre tres instancias reales de complejidad creciente para evaluar la capacidad de búsqueda de los algoritmos, algo que no se encuentra tampoco en la literatura.

8.1.2. Búsqueda dispersa

Hasta donde conocemos, no existe ningún trabajo que utilice esta técnica para resolver el problema de la planificación de celdas. La novedad reciente de la búsqueda dispersa y su aplicación preferente en el campo de la investigación operativa pueden explicar este hecho. Esta tesis supone una importante contribución en este sentido.

8.2. Representación y operadores utilizados

Los esquemas generales de los tres algoritmos utilizados para resolver el problema ACP, léase ssNSGA-II, AbYSS y pPAES ya se presentaron en el Capítulo 7, dejando sólo por determinar todos los aspectos relativos al problema en cuestión que se pretende abordar. Así, esta sección incluye la representación de las soluciones y los operadores utilizados por los tres algoritmos. Es importante recordar que se han utilizado los mismos operadores estocásticos típicos del campo de los algoritmos evolutivos dentro del esquema de la búsqueda dispersa. Aunque en teoría este tipo no son apropiados para el esquema de SS, procedente del campo de investigación operativa, existen trabajos en la literatura donde su uso mejora considerablemente la capacidad de búsqueda del algoritmo, tanto para problemas monoobjetivo [110] como multiobjetivo [192]. Esta aproximación no sólo se sigue para el problema ACP, sino también para los otros dos problemas abordados en esta tesis.

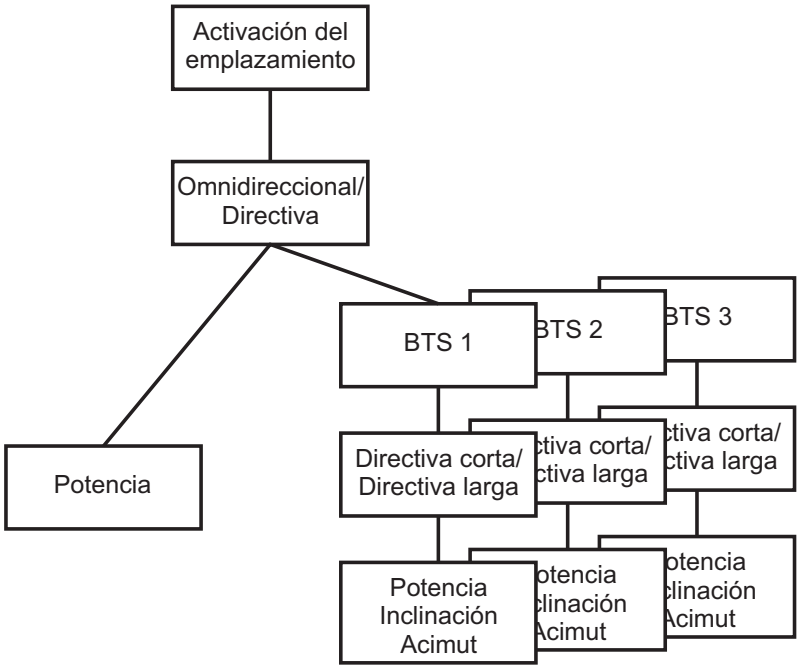


Figura 8.2: Codificación jerárquica de la red.

8.2.1. Codificación de las soluciones

Las soluciones que manipulan los algoritmos considerados codifican la red completa que se pretende optimizar (localizaciones, número de antes antenas y parámetros de configuración). Se ha utilizado una codificación multinivel en la que el nivel 1 indica si el emplazamiento está activado o no, el nivel 2 hace referencia el número y tipo de BTSs instaladas en el emplazamiento y el nivel 3 fija los parámetros de estas BTSs. Así, cuando un emplazamiento está activado, una o más antenas se activan, manteniendo siempre una única omnidireccional o de 1 a 3 direccionales. La Figura 8.2 muestra la codificación jerárquica utilizada en las soluciones. Como se puede observar, no se trata de una codificación clásica, por lo que los operadores que utilizan los algoritmos han de ser diseñados adecuadamente.

8.2.2. Operadores genéticos

Aunque no puede considerarse un operador como tal, la generación de la población inicial también depende en gran medida del problema concreto que se está abordando, de ahí su inclusión en esta sección. Hemos utilizado una estrategia totalmente aleatoria para esta generación inicial de soluciones: por cada emplazamiento de la red, se decide si se activa o no. Si es así, se genera entonces una configuración que sigue la codificación por niveles de la sección anterior, es decir, se determina primero si se instala una BS omnidireccional o varias direccionales. En el primer caso, se escogen aleatoriamente dos valores de entre el conjunto de valores discretizados (ver Tabla 4.1) para la potencia de la BTS. Si las BTSs son direccionales, hay que determinar cuántas se instalan. Recordemos que este valor ha de estar entre 1 y 3. Entonces, por cada BTS direccional hay que establecer su diagrama de propagación (directiva corta o directiva larga), así como los valores de potencia, inclinación y acimut (Tabla 4.1). En todos los casos, los valores de la configuración se eligen de forma aleatoria utilizando una distribución uniforme de probabilidad.

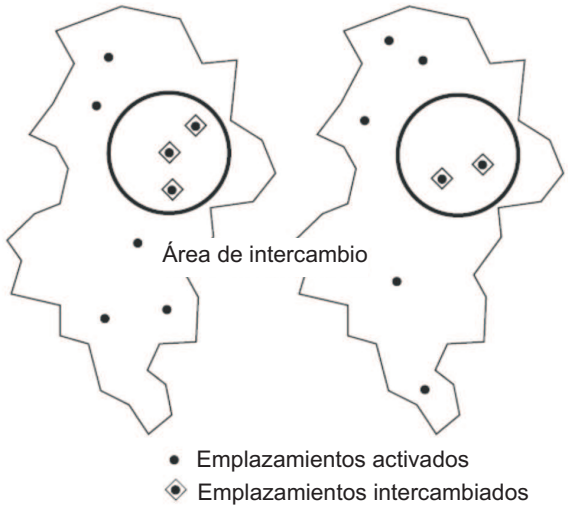


Figura 8.3: Recombinación geográfica. Los emplazamientos localizados dentro del radio de acción son intercambiados.

Pasemos ahora a describir los operadores de recombinación y mutación. Hay que tener en cuenta que estos operadores son utilizados no sólo por ssNSGA-II y pPAES (sólo mutación), sino también son la base para los métodos de mejora y de combinación de soluciones de AbYSS (ver siguiente sección). Se denominan recombinación geográfica y mutación multinivel, respectivamente.

La recombinación geográfica trabaja en el nivel 1 de la representación jerárquica y está basada en el intercambio de la configuración de un área de la red que está en dos soluciones (individuos) distintas. En primer lugar, se eligen aleatoriamente un emplazamiento de la red (L_i) y un valor aleatorio que representa el radio del área a intercambiar. Entonces, todos los emplazamientos de la red que están dentro de este radio de acción con respecto al emplazamiento aleatorio elegido (con respecto a L_i) se intercambian entre las dos soluciones, con toda la información presente en los niveles 2 y 3 de la codificación, es decir, las características de propagación de las antenas.

La mutación multinivel, por su parte, puede actuar en todos los niveles de la codificación de los individuos. Cada vez que una solución se muta, sólo un único nivel se modifica. El operado trabaja como sigue. En primer lugar, se elige aleatoriamente el emplazamiento, L_i, que sufrirá la mutación. Una vez seleccionado, existen cinco posibles modificaciones que se pueden realizar en L_i:

- Conmutación activado/desactivado. Si L_i está activado, entonces se desactiva. Si, por el contrario, s está desactivado, entonces se genera una configuración aleatoria para L_i (una antena omnidireccional o varias direccionales, potencia de emisión, ángulos de inclinación y acimut, diagramas de radiación, etc.).

- Actualización de la potencia de emisión. Necesita que L_i esté activado. Elige aleatoriamente una BTS de L_i, para después decidir, también aleatoriamente, entre uno de los dos valores posibles de potencia adyacentes (inferior y superior) de la BTS.

- Actualización del ángulo de inclinación. Es igual que modificar la potencia, pero únicamente es aplicable a las BTSs direccionales (que tienen definido este ángulo). También se elige entre los valores válidos adyacentes (inferior y superior) del ángulo de inclinación de la BTS.

- Actualización del ángulo de acimut. Se trata de una actualización similar a la anterior (bajo las mismas condiciones) pero modificando el ángulo de acimut.

- Actualización del diagrama de radiación. Al igual que las tres actualizaciones anteriores, aquí se necesita que L_i esté activado. Lo que se pretende es cambiar el tipo de BTS, de forma que se pasa de tener una BTS omnidireccional a una o varias BTS directivas, y viceversa. La nueva configuración de las BTSs se genera aleatoriamente.

Nótese que ambos operadores siempre producen soluciones válidas, es decir, soluciones que cumplen todas las restricciones de diseño impuestas por la red (i.e., de 1 a 3 BTSs direccionales, valores discretos para potencia, inclinación y acimut).

8.2.3. Métodos de AbYSS

De los cinco métodos que definen la plantilla de la búsqueda dispersa, los de generación de soluciones diversas, de mejora y de combinación de soluciones se definen en base al problema que se está resolviendo. El método de actualización del conjunto de referencia, además, utiliza el concepto de distancia entre soluciones, que también depende del problema de optimización concreto. Como ya se ha mencionado anteriormente, hay métodos de AbYSS que están basados en los operadores estocásticos utilizados en ssNSGA-II y pPAES, en concreto, los de mejora y combinación de soluciones. Esto nos va a permitir comparar, en cierta medida, la adecuación de los motores de búsqueda de los tres algoritmos para resolver el problema. A continuación se describen de forma separada cada uno de estos métodos.

Generación de soluciones diversas

Para este método concreto no hemos diseñado ninguna estrategia específica para generar soluciones diversas debido a la complejidad propia de las soluciones. Esto hace que soluciones generadas aleatoriamente (de la misma forma que en los EAs) cubran, de forma efectiva, el espacio de búsqueda. No obstante, se podrían definir esquemas en los que esta generación aleatoria esté dirigida, por ejemplo, a tener diferentes porcentajes de emplazamientos activados, una proporción determinada entre BTS omnidireccionales y directivas, etc.

Mejora

En el diseño original de AbYSS [192], el método de mejora consiste en una estrategia (1+1) EA que está basada en un operador de mutación junto con un test de dominancia de Pareto. La misma aproximación se ha seguido aquí, sólo que, en este caso, el operador de mutación es la mutación multinivel presentada en la Sección 8.2.2. El esquema de este método de mejora se muestra en el Algoritmo 8.1.

El funcionamiento consiste en, dada una solución que se pasa como parámetro, se muta *iter* número de veces intentando conseguir mejores soluciones. El término "mejor" aquí se define de la misma forma que hace NSGA-II [67]. Primero, un test de violación de restricciones comprueba la validez de las dos soluciones (línea 6). Si una cumple las restricciones y la otra no, o ambas no las cumplen pero una de ellas las viola en una cantidad menor, el test devuelve la ganadora (línea 7). En otro caso, se utiliza un test de dominancia para determinar qué solución domina a la otra. Así, si la solución original gana, la mutada se descarta. Si es la mutada la que domina, entonces reemplaza a la original. Finalmente, si ambas son no dominadas y, además, la solución mutada no está dominada por el archivo externo, la solución original se envía al archivo y la mutada se queda como la original. Así se evita que se obtengan peores soluciones respecto al archivo al iterar el método.

Algoritmo 8.1 Pseudocódigo que describe el método de mejora.

```
1: entrada: solución tentativa s, número de iteraciones t
2: solucionOriginal ← s
3: for (i = 0; i <t; i++) do
4:    solucionMutada ← mutacionMultinivel(solucionOriginal)
5:    if el problema tiene restricciones then
6:       evaluarRestricciones(solucionMutada)
7:       mejor ← testDeRestricciones(solucionMutada, solucionOriginal)
8:       if ninguno es mejor que el otro then
9:          evaluar(solucionMutada)
10:         mejor ← testDeDominancia(solucionMutada, solucionOriginal)
11:      else if solucionMutada es mejor then
12:         evaluar(solucionMutada)
13:      end if
14:   else
15:      // el problema no tiene restricciones
16:      evaluar(solucionMutada)
17:      mejor ← testDeDominancia(solucionMutada, solucionOriginal)
18:   end if
19:   if solucionMutada es mejor then
20:      solucionOriginal ← solucionMutada
21:   else if solucionMutada no está dominada por el archivo externo then
22:      //ambas soluciones son no dominadas
23:      insertar solucionOriginal en el archivo externo
24:      solucionOriginal ← solucionMutada
25:   end if
26: end for
27: return solucionOriginal
```

Combinación de soluciones

La idea de este método es, dado un par de soluciones s_1 y s_2 procedentes del método de generación de subconjuntos, generar nuevas soluciones que combinen adecuadamente la información contenida en ambas. En AbYSS, diseñado originalmente para funciones de variable real, el método utilizado ha sido el cruce SBX [66]. En este caso, para el problema ACP hemos utilizado la recombinación geográfica descrita en la Sección 8.2.2.

Medida de distancia para el método de actualización del $RefSet$

Si bien el método en sí quedó definido en el Capítulo 7, dada la representación de soluciones utilizada para resolver el problema, hay que definir qué se entiende por distancia entre dos posibles configuraciones de una red dada. Sea $L = \{L_1, L_2, \ldots, L_n\}$ el conjunto de emplazamientos de la red y sean s_1 y s_2 dos posibles configuraciones para esa red (soluciones tentativas). Así, $L_i^{s_j}$ es la configuración del emplazamiento L_i en la solución s_j. Partiendo de un valor $d = 0$, la distancia se calcula como sigue. Para todo emplazamiento $L_i \in L$:

1. Si ni $L_i^{s_1}$ ni $L_i^{s_2}$ están activados, entonces: $d \leftarrow d + 0$. Es decir, el valor de d no cambia.
2. Si uno está activado y otro no, entonces $d \leftarrow d + 10$. Sea, sin perdida de generalidad, que $L_i^{s_1}$ es el emplazamiento activado. Entonces, además:

a) Si $L_i^{s_1}$ tiene instalado una antena omnidireccional: $d \leftarrow d + 2$. Sumamos una unidad por cada parámetro adicional. En este caso, la antena omnidireccional tiene sólo 1: la potencia.

b) Si el emplazamiento activado, $L_i^{s_1}$, tiene instaladas varias antenas direccionales, la distancia se incrementa en 4 unidades por cada una de estas antenas, es decir, $d \leftarrow d + 4 \times \text{n}^\text{o}\_\text{de}\_\text{antenas}$ (las antenas direccionales tienen 4 parámetros, el tipo, la potencia y los ángulos de inclinación y acimut).

3. Si $L_i^{s_1}$ y $L_i^{s_2}$ están ambos activados, entonces la distancia dependerá del tipo de BTSs instaladas:

a) Si ambas configuraciones equipan los emplazamientos con una única BTS omnidireccional y emiten a distinta potencia, entonces $d \leftarrow d+1$. En otro caso, el valor de la distancia no se modifica. Hay que tener en cuenta que la configuración de estos emplazamientos es muy similar y, por tanto, tiene sentido que la distancia entre ambos sea pequeña.

b) Si las antenas son directivas en las dos configuraciones, entonces:

1) Con el mismo número de BTSs, se suma 1 por BTS si existe un parámetro de configuración con un valor diferente. Así $d \leftarrow d + k$, $0 \leq k \leq 3$, en este caso.
2) Si hay un número distinto de BTSs instaladas, la distancia se incrementa en 1 por cada una de ellas que tenga un valor diferente en algún parámetro de configuración, y se suma 4 por cada BTS que esté en un emplazamiento y no en otro.

c) Si una configuración tiene instalada una antena omnidireccional y la otra tiene una o varias antenas directivas, entonces:

1) Cuando hay tres antenas directivas, $d \leftarrow d + 3$
2) Cuando hay dos antenas directivas, $d \leftarrow d + 6$
3) Cuando hay una antena direccional, $d \leftarrow d + 9$

es decir, se consideran mucho más similares aquellas las configuraciones que tienen tres antenas direccionales, ya que las celdas tienden a ser más parecidas.

La idea tras esta medida de distancia desarrollada *ad hoc* es que tenga valores más altos para configuraciones en las que los emplazamientos son muy distintos: estén o no activados, antena omnidireccional vs. varias antenas directivas, etc.

8.3. Instancias abordadas

En esta sección se presentan las tres instancias del problema ACP que han sido abordadas en esta tesis. Se trata de datos reales de redes de telefonía móvil que han sido proporcionadas por France Telecom, que se corresponden con dos tipos de escenarios bien diferenciados: la instancia Arno 1.0 es una autovía, mientras que las instancias Arno 3.0 y Arno 3.1 son dos zonas urbanas de distinto tamaño. La topología de estas tres instancias aparecen, respectivamente, en las Figuras 8.4, 8.5 y 8.6.

La Tabla 8.4 incluye algunos de los valores más importantes que caracterizan las tres instancias. Se puede observar que el tamaño es creciente, pasando de los 250 emplazamientos candidatos posibles en Arno 1.0 a 743 (casi el triple) de la instancia Arno 3.1. Hay que tener en cuenta que no sólo se incrementa el número de emplazamientos, sino también los puntos de test utilizados para evaluar la calidad del servicio que es capaz de proporcionar la red. A modo de ejemplo, el tiempo de evaluación media de cada una de estas instancias en un procesador AMD Opteron a 2,2 GHz, es de 6, 5 y 22 segundos, respectivamente. Así, para una ejecución típica de 25.000 evaluaciones

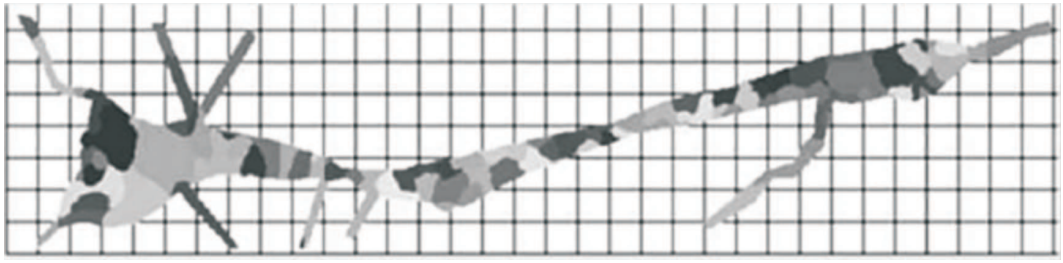


Figura 8.4: Instancia Arno 1.0.

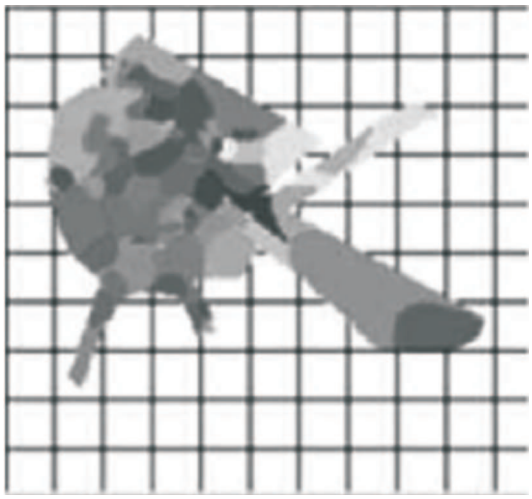


Figura 8.5: Instancia Arno 3.0.

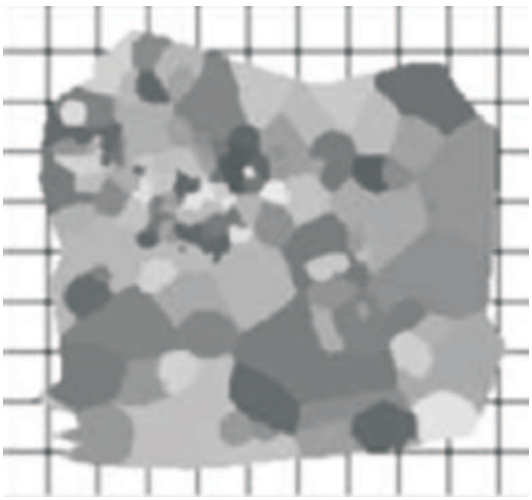


Figura 8.6: Instancia Arno 3.1.

Instancia	Arno 1.0	Arno 3.0	Arno 3.1
Tráfico total (Erlangs)	3.210,91	2.988,12	8.089,78
Número de emplazamientos	250	568	747
Número de puntos de test de servicio ($\|\mathcal{ST}\|$)	29.955	17.394	42.975
Número de puntos de test de tráfico ($\|\mathcal{T}\|$)	4.967	6.656	21.475

Tabla 8.4: Algunos valores que caracterizan las tres instancias resueltas.

de función, los algoritmos tardarán, sin contar con su propia sobrecarga de cómputo, 41, 34 y 152 horas. Esto pone de manifiesto la necesidad de utilizar estrategias paralelas para abordar el problema.

8.4. Experimentos

En esta sección se presenta la experimentación realizada para abordar las tres instancias planteadas del problema de la planificación de celdas. En la Sección 8.4.1 se describen los algoritmos de optimización utilizados, así como la configuración de cada uno de ellos. Los resultados se han dividido en tres apartados. En primer lugar, hemos comparado nuestras propuestas algorítmicas para este problema (ssNGSA-II y AbYSS) con los dos algoritmos del estado del arte, NSGA-II [68] y SPEA2 [262]. En esta comparativa hemos incluido también a PAES [136], ya que es el algoritmo base de pPAES, nuestra tercera propuesta para resolver este problema. El siguiente paso ha sido evaluar pPAES no sólo desde un punto de vista de calidad de soluciones, sino también desde el punto de vista del paralelismo. Para terminar, esta sección concluye con los resultados obtenidos por las extensiones grid de nuestras propuestas.

Para medir la calidad de los frentes obtenidos por los distintos algoritmos hemos utilizado el indicador Hipervolumen (Sección 3.5), ya que no necesita conocer previamente el frente óptimo del problema, como es nuestro caso. Queremos recordar en este punto que los resultados que se presentan en las siguientes secciones están promediados sobre 30 ejecuciones independientes, lo que supone un enorme esfuerzo computacional teniendo en cuenta el tiempo de cómputo necesario para realizar las 25.000 evaluaciones de cada instancia considerada (ver sección anterior).

8.4.1. Configuración de los algoritmos

En esta sección se incluye la parametrización de todos los algoritmos utilizados para resolver este problema. Para hacer una comparación apropiada de los motores de búsqueda de los algoritmos (recordemos que todos utilizan los mismos operadores), todas las propuestas utilizan la misma condición de parada: 25.000 evaluaciones de funciones; y calculan un frente de tamaño máximo 100, es decir, como mucho pueden obtener 100 soluciones no dominadas. Partiendo de esta base y, después de realizar experimentación preliminar, la configuración detallada de cada algoritmo utilizado en esta sección es la siguiente:

- **ssNSGA-II**. Nuestra propuesta utiliza una población de 100 individuos. Los operadores de recombinación y mutación (Sección 8.2.2) se aplican respectivamente con probabilidad $p_c = 0{,}9$ y $p_m = 1/L$, donde L es el número de emplazamientos de la red.
- **AbYSS**. Hemos utilizado un tamaño de 20 para la población inicial P y de 10 para $RefSet_1$ y $RefSet_2$. Para los métodos de combinación de soluciones y mejora, los operadores de recombinación y mutación utilizados se aplican siempre ($p_c = p_m = 1{,}0$), aunque en el caso de la mutación sólo se cambia un emplazamiento cada vez. Además, el número de iteraciones del método de mejora es 1.

- **PAES**. Este algoritmo utiliza la mutación multinivel como operador de búsqueda. Al igual que el método de mejora de AbYSS, se aplica siempre, pero se modifica sólo un emplazamiento a la vez. La rejilla adaptativa que utiliza PAES para incluir diversidad en el frente tiene un número de subdivisiones máxima de 5.

- **NSGA-II** y **SPEA2**. Ambos utilizan las mismas probabilidades de cruce y mutación que ssNSGA-II, es decir, $p_c = 0{,}9$ y $p_m = 1/L$, donde L es el número de emplazamientos de la red. La población de SPEA2, además, es de 100 individuos.

8.4.2. Resolución con los algoritmos propuestos

Los primeros resultados que incluimos en este capítulo miden la calidad de las soluciones obtenidas por ssNSGA-II, AbYSS y PAES, así como por los algoritmos del estado del arte, NSGA-II y SPEA2. Dado que estamos utilizando los mismos operadores de búsqueda en todos ellos, estos resultados van a permitir comparar la adecuación de los distintos motores de búsqueda para resolver el problema ACP.

La Tabla 8.5 muestra la mediana, $\tilde{x}$, y el rango intercuartílico, IQR, sobre 30 ejecuciones independientes para el indicador Hipervolumen (HV) alcanzado por los cinco algoritmos considerados en las tres instancias Arno 1.0, Arno 3.0 y Arno 3.1. Las celdas de la tabla que tienen fondo gris muestran los mejores (mayores) valores para este indicador de calidad. Nótese que hemos utilizado la mediana y rango intercuartílico como índices de centralización y de dispersión, en lugar de medias y desviaciones típicas, ya que no todas las muestras siguen una distribución normal. Por su parte, la Tabla 8.6 incluye el resultado de los tests de comparaciones múltiples entre cada par de algoritmos y por cada instancia.

El primer análisis con el que empezaremos esta discusión va a estar centrado en comparar nuestras tres propuestas, ssNSGA-II, AbYSS y PAES. Como se puede observar en la Tabla 8.5, ssNSGA-II es el algoritmo que alcanza los frentes de Pareto con mayor valor (mejor) de HV para las tres instancias y, además, con significancia estadística (ver los símbolos "+" en la Tabla 8.6). Es posible dar una justificación para estos resultados. En el caso de AbYSS, la explicación tiene que ver con el tamaño de las instancias y el funcionamiento propio de la técnica, en concreto, con su fase de reinicio (Sección 7.2.2). En este algoritmo, esta fase de reinicio resulta fundamental ya que permite realimentar la búsqueda con soluciones no dominadas del frente de Pareto encontrado hasta el momento, con el objetivo de intensificar la exploración de esa región del espacio de búsqueda. No obstante, el tamaño de las instancias, junto con los operadores y la medida de distancia utilizados, hacen que en el bucle interno de la búsqueda dispersa se encuentren frecuentemente soluciones que se satisfacen las condiciones para su inclusión en el $RefSet_2$. Esto provoca que el reinicio se realice en pocas ocasiones, reduciendo así esta realimentación. En cuanto a PAES, los resultados son de

Tabla 8.5: Hipervolumen obtenido por ssNSGA-II, AbYSS, PAES, NSGA-II y SPEA2 para las tres instancias del problema ACP.

	Arno 1.0	Arno 3.0	Arno 3.1
Algoritmo	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$	$\tilde{x}_{IQR}$
ssNSGA-II	$0{,}5833_{0{,}0217}$	$0{,}5734_{0{,}0081}$	$0{,}4776_{0{,}0094}$
AbYSS	$0{,}4569_{0{,}0296}$	$0{,}4469_{0{,}0227}$	$0{,}3602_{0{,}0292}$
PAES	$0{,}4504_{0{,}0819}$	$0{,}2519_{0{,}0336}$	$0{,}1254_{0{,}0311}$
NSGA-II	$0{,}5763_{0{,}0369}$	$0{,}5627_{0{,}0166}$	$0{,}4731_{0{,}0180}$
SPEA2	$0{,}5497_{0{,}0268}$	$0{,}5364_{0{,}0219}$	$0{,}4600_{0{,}0198}$

Tabla 8.6: Resultados del test de comparaciones múltiples.

AbYSS	1.0	+										
	3.0		+									
	3.1			+								
PAES	1.0	+			−							
	3.0		+			+						
	3.1			+			+					
NSGA-II	1.0	−			+			+				
	3.0		−			+			+			
	3.1			−			+			+		
SPEA2	1.0	+			+			+			+	
	3.0		+			+			+			+
	3.1			+			+			+		+
	Arno	1.0	3.0	3.1	1.0	3.0	3.1	1.0	3.0	3.1	1.0	3.0	3.1
		ssNSGA-II			AbYSS			PAES			NSGA-II		

alguna forma los esperados puesto que, al no utilizar recombinación alguna de las soluciones, limita que se explore adecuadamente el complejo espacio de búsqueda del problema.

Gráficamente, los resultados de los algoritmos se pueden ver en las Figuras 8.7, 8.8 y 8.9, que muestran, cada una, tres frentes de Pareto típicos de ssNSGA-II, AbYSS y PAES para las instancias Arno 1.0, Arno 3.0 y Arno 3.1, respectivamente. En todos los casos se puede observar cómo ssNSGA-II obtiene los frentes más próximos a las zonas de alta calidad (a la derecha de cada figura, donde el tráfico es máximo, las interferencias mínimas y el número de emplazamientos activados es el más bajo), dominando a los producidos por AbYSS y PAES. Las figuras también ponen de manifiesto las debilidades de estos dos algoritmos comentadas en el párrafo anterior. Por su parte, la falta de realimentación provoca que AbYSS no converja tanto como ssNSGA-II en las tres instancias, mientras que en el caso de PAES, la falta de recombinación hace que aparezcan soluciones aisladas, que se han generado en algún momento de la búsqueda, y cuya información no ha sido capaz de incorporar el algoritmo para explorar otras regiones del espacio de soluciones. Es importante reseñar también que, aunque la forma de los frentes para las tres instancias es parecida, los objetivos se mueven en escalas distintas. Nótese el amplio rango de configuraciones de red que proporcionan los algoritmos, especialmente ssNSGA-II y AbYSS. Ambos ofrecen al diseñador soluciones que van desde un número muy bajo de emplazamientos activados (y, por tanto, de bajo coste) que, por supuesto, conllevan también interferencias mínimas pero también pueden dar servicio a una escasa cantidad de tráfico, a configuraciones más costosas por el número de antenas involucradas, pero que soportan más tráfico (y también con más interferencias). PAES, por el contrario, se suele concentrarse en zonas muy concretas del espacio de soluciones.

Está claro que ssNSGA-II es el que proporciona mejores frentes de entre nuestras tres propuestas. No obstante, si nos fijamos tanto en los valores de HV que alcanza como en los frentes de Pareto que es capaz de obtener, creemos que los resultados de AbYSS son muy prometedores ya que es la primera vez que un algoritmo basado en búsqueda dispersa se aplica a este tipo de problemas. Así, si tomamos como referencia los valores de HV de ssNSGA-II (Tabla 8.5), se puede observar que el HV de los frentes de AbYSS para las instancias Arno 1.0, Arno 3.0 y Arno 3.1 es, respectivamente, el 78 %, 77 % y 75 % de los alcanzados por ssNSGA-II. Esto es, se mantiene más o menos constante con la complejidad creciente de las instancias o, lo que es lo mismo, se puede concluir que AbYSS escala bien para este problema. El diseño de nuevas medidas de distancia y, quizás, de un operador de búsqueda local más intensiva pueden conducir a mejoras considerables en el rendimiento de este algoritmo. Las diferencias en los valores de HV de PAES con respecto

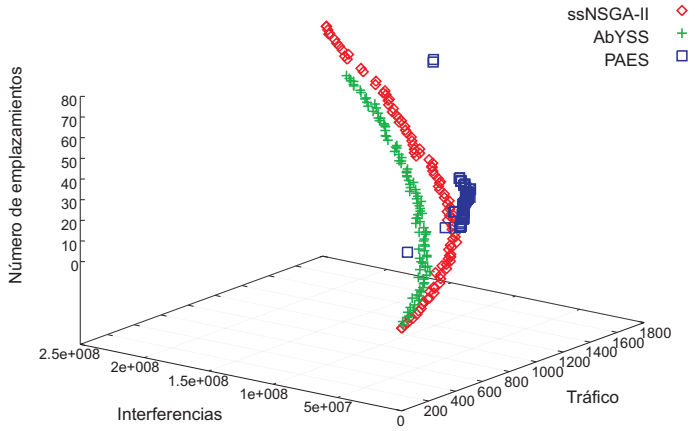


Figura 8.7: Frentes de Pareto de ssNSGA-II, AbYSS y PAES para la instancia Arno 1.0.

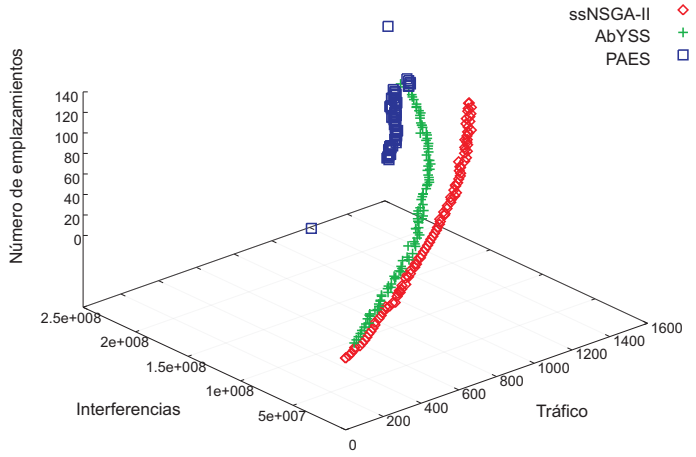


Figura 8.8: Frentes de Pareto de ssNSGA-II, AbYSS y PAES para la instancia Arno 3.0.

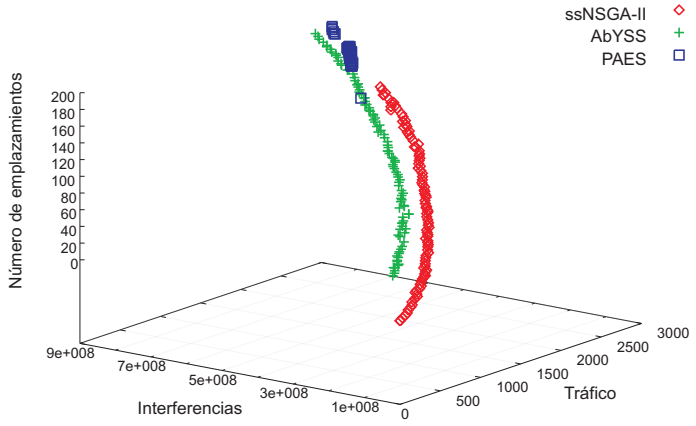


Figura 8.9: Frentes de Pareto de ssNSGA-II, AbYSS y PAES para la instancia Arno 3.1.

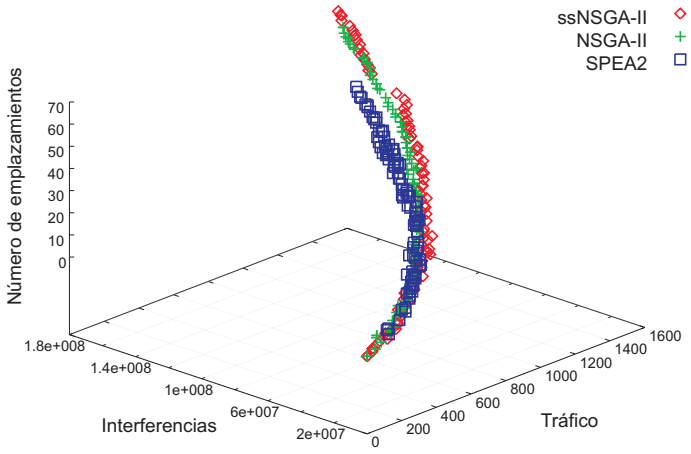


Figura 8.10: Frentes de Pareto de ssNSGA-II, NSGA-II y SPEA2 para la instancia Arno 1.0.

a los de ssNSGA-II, por el contrario, son del 77 %, 43 % y 26 % para Arno 1.0, Arno 3.0 y Arno 3.1, respectivamente. Estos resultados dejan patente la escasa escalabilidad de PAES al abordar las instancias más grandes. A pesar de esto, el rendimiento de PAES en la instancia Arno 1.0 es similar al de AbYSS según el indicador HV, no habiendo diferencias significativas entre ambos algoritmos a un nivel de confianza del 95 % (ver Tabla 8.6). Para interpretar este resultado hay que recordar que el indicador HV mide tanto convergencia hacia el frente óptimo como diversidad de las soluciones no dominadas. La situación se muestra en la Figura 8.7, donde se puede observar que, si bien AbYSS ha obtenido un conjunto de soluciones que cubre un espectro más grande de configuraciones, PAES ha sido capaz de alcanzar soluciones más cercanas a la región óptima. Así, traducido a valores de HV, la convergencia de PAES se compensa con la diversidad de AbYSS.

Introduzcamos ahora en el análisis los resultados de los dos algoritmos de referencia, NSGA-II y SPEA2. Como se puede observar en la Tabla 8.5, nuestra propuesta, ssNSGA-II es el algoritmo que obtiene los frentes de Pareto con mejor valor en el indicador de calidad HV para las tres instancias, también cuando se consideran estos dos algoritmos. Los resultados del test de comparaciones múltiples (Tabla 8.6) indican, además, que las diferencias son significativas entre ssNSGA-II y SPEA2, pero no lo son para ssNSGA-II y NSGA-II utilizando un nivel de 95 % de confianza. Aunque estas diferencias de HV son pequeñas en valor absoluto, no hay que olvidar que los valores que aparecen en la Tabla 8.5 están calculados después de que los diferentes frentes hayan pasado por un proceso de normalización que previene de la presencia de resultados engañosos. De hecho, en muchos de los frentes alcanzados por los tres algoritmos en las tres instancias, la situación es la que aparece en la Figura 8.10. Tanto ssNSGA-II como NSGA-II son capaces de llegar a configuraciones para la red con valores muy similares de número de emplazamientos, tráfico e interferencias, pero el primero converge algo más que el segundo (es decir, soluciones de más calidad), consecuencia de la selección por estado estacionario. SPEA2, por su parte, encuentra algunas dificultades no sólo para cubrir la misma región del espacio de soluciones que ssNSGA-II y NSGA-II, sino también en cuanto a convergencia hacia el frente óptimo. Para completar este análisis, también es necesario indicar que tanto AbYSS como PAES, nuestras otras dos propuestas para resolver el problema ACP, obtienen un conjunto de soluciones no dominadas con HV más bajo (peor) que NSGA-II y SPEA2.

Llegado este punto podemos extraer varias conclusiones atendiendo a los resultados de los experimentos realizados. En primer lugar, el motor de búsqueda de los EAs que incorporan operador de recombinación (ssNSGA-II, NSGA-II y SPEA2) es el que mejor adecua a este problema. Así, en el caso de PAES, donde no se utiliza recombinación, la exploración del espacio de búsqueda no es

tan efectiva y tiende a quedarse estancada en zonas concretas del espacio de objetivos. AbYSS, por su parte, sí que utiliza recombinación de soluciones y esto le permite cubrir regiones amplias de los frentes, así como escalar con respecto al tamaño de la instancia considerada, pero aún necesita un ajuste mayor para mejorar su funcionamiento (e.g., nuevas medidas de distancia entre soluciones).

Para terminar esta sección, aunque no estamos comparando aquí los tiempos de ejecución de los algoritmos, queremos mostrar el tiempo que se necesita para obtener los resultados presentados en la Tabla 8.5. Si tenemos en cuenta los tiempos que mostrados al final de la Sección 8.3 y que hemos realizado 30 ejecuciones independientes de cada algoritmo, tenemos que son necesarias 150 pruebas por cada instancia con lo que, en media, en un AMD Opteron 2,2, tendríamos: $150 \times (41 + 34 + 152)$ = 1418 horas = 3,88 años de tiempo aproximado de cómputo. En nuestro caso hemos utilizados una gran cantidad recursos (diferentes clusters dedicados del grupo de investigación, así como laboratorios del Departamento de Lenguajes y Ciencias de la Computación) para poder obtener los resultados en un tiempo razonable. No obstante, queda claro que la utilización de técnicas paralelas es casi obligatorio para abordar este problema. En este sentido, el esquema de ssNSGA-II nos va a permitir aprovechar eficientemente la potencia de cómputo que proporcionan los sistemas de computación grid compuestos por cientos de procesadores, algo que las versiones originales de NSGA-II y SPEA2 con esquema generacional no pueden conseguir. Hemos diseñado, por tanto, un algoritmo que resuelve mejor el problema ACP que los algoritmos de referencia y que, además, nos va a permitir reducir el tiempo de ejecución de varios días a menos de dos horas, como se verá más adelante, en la Sección 8.4.4.

8.4.3. Efectividad y eficiencia de pPAES

Esta sección está dedicada a evaluar pPAES. Debido la escasa efectividad de PAES en la sección anterior para las instancias Arno 3.0 y Arno 3.1, este estudio se ha centrado únicamente en la instancia Arno 1.0. Además, en lugar de utilizar como condición de parada la computación de 25.000 evaluaciones de función, se ha considerado llegar sólo a 10.000. Este valor es lo suficientemente grande como para mostrar las características paralelas de nuestra propuesta.

Se han probado tres configuraciones distintas de pPAES: pPAES_4, pPAES_8 y pPAES_{16} en las que se utilizan, respectivamente 4, 8 y 16 procesos paralelos, configurados siempre con una topología en un anillo unidireccional. Cada subalgoritmo se ejecuta en un procesador dedicado, por lo que hemos utilizado una plataforma paralela compuesta por hasta 16 máquinas. Cada una de estas máquinas es un Pentium 4 a 2,8 GHz con 512 MB de RAM y están interconectadas mediante una Fast-Ethernet a 100 Mbps. La mutación multinivel se aplica de la misma forma que en PAES, es decir, se muta siempre, pero en cada ocasión sólo se modifica un emplazamiento. Cada una de las islas tiene un archivo local de soluciones no dominadas con tamaño máximo 100. Finalmente, la operación de migración de soluciones no dominadas se produce cada 30 iteraciones.

Comencemos estudiando el comportamiento paralelo del algoritmo. La Tabla 8.7 incluye los tiempos medios de ejecución de PAES y de las tres configuraciones de pPAES, sobre 30 ejecuciones independientes, para la instancia Arno 1.0, así como la eficiencia paralela, η, que se consigue. Lo

Tabla 8.7: Tiempos de ejecución (en segundos) y eficiencia paralela de los algoritmos para el problema Arno 1.0.

Algoritmo	Tiempo (s)	η
PAES	19.777	–
pPAES_4	5.153	95,95 %
pPAES_8	3.277	75,44 %
pPAES_{16}	2.160	57,23 %

Tabla 8.8: Indicador HV alcanzado por las diferentes configuraciones de pPAES en el problema Arno 1.0.

	HV
Algoritmo	$\bar{x}_{\pm\sigma_n}$
PAES	$0{,}2590_{\pm 0{,}0351}$
$pPAES_4$	$0{,}2851_{\pm 0{,}0285}$
$pPAES_8$	$0{,}2682_{\pm 0{,}0305}$
$pPAES_{16}$	$0{,}2535_{\pm 0{,}0285}$

Tabla 8.9: Resultados del test de comparaciones múltiples con las diferentes configuraciones de pPAES.

$pPAES_4$	+		
$pPAES_8$	−	−	
$pPAES_{16}$	−	+	−
	PAES	$pPAES_4$	$pPAES_8$

primero que se observa es que, efectivamente, se consigue reducir el tiempo de cómputo a medida que aumentamos el número de procesadores. De hecho, $pPAES_4$ ha obtenido un valor casi óptimo de η (95,95 %). En el caso de $pPAES_8$, la eficiencia paralela es del 75,44 % que es un valor más bajo, pero aceptable. Esta reducción en la eficiencia paralela (acentuada en $pPAES_{16}$, donde cae hasta el 57,23 %) era esperada debido a las sincronía introducida por el esquema de pPAES. Como ya se explicó, el anillo unidireccional que utiliza este algoritmo como topología de comunicación es síncrono. Esto, junto con el hecho de que el tiempo de evaluación de una red es variable (depende en gran medida del número de emplazamientos activados), hace que haya procesos sin computar. Además, el tamaño en bytes de la instancia Arno 1.0 es aproximadamente de 300 KB, lo que hace aún más desfavorable el ratio comunicación/computación. Esta característica es especialmente perjudicial en la fase de finalización de pPAES, en la que un proceso distinguido se encarga de recoger todos los frentes de Pareto del resto (normalmente de tamaño 100) de subalgoritmos para obtener el conjunto de soluciones no dominadas final. Por ejemplo, si tenemos en cuenta $pPAES_{16}$, entonces 15 subalgoritmos $\times$ 100 soluciones $\times$ 300 KB $\simeq$ 439 MB. Nótese que hemos utilizado 15 procesos ya que el nodo distinguido no tiene que enviar su frente por la red. Transferir esta cantidad de información por la red también supone un tiempo considerable que penaliza la eficiencia paralela de pPAES a medida que crece el número de islas.

Para medir la calidad de las soluciones que obtiene pPAES, hemos usado nuevamente el indicador HV (Tabla 8.8). Los valores presentados son la media, $\bar{x}$, y la desviación típica, σ_n, de 30 ejecuciones independientes, ya que todas las muestras siguen ahora una distribución normal. Aquí los resultados son muy prometedores en lo que al modelo de búsqueda se refiere. Como se puede observar, tanto $pPAES_4$ como $pPAES_8$ han alcanzado un valor de HV más alto que el de PAES y con significancia estadística en el primer caso, como indica el símbolo "+' en la Tabla 8.9. Con respecto a $pPAES_{16}$, PAES es sólo ligeramente mejor (0,2590 frente a 0,2535). Estos datos nos vienen a decir que la capacidad de búsqueda de pPAES supera a la de PAES para este problema. Es decir, la colaboración entre diferentes PAES ha permitido encontrar un conjunto de soluciones no dominadas que supera en convergencia y diversidad al frente de Pareto que alcanza PAES. Estos valores de HV, junto con la reducción de tiempo que se puede conseguir al ser pPAES una metaheurística paralela, ponen de manifiesto que hemos diseñado una metaheurística que puede resultar muy apropiada para resolver este problema.

8.4.4. Resultados en sistemas de computación grid

En esta sección se presentan los resultados de aplicar assNSGA-II (*asynchronous steady state NSGA-II*) a la instancia Arno 3.1. Se ha seleccionado esta instancia porque es la que supone una mayor carga computacional y, por tanto, unos tiempos de ejecución más elevados (del orden de varios días). La parametrización de assNSGA-II es exactamente la misma que la de ssNSGA-II (Sección 8.4.1), sólo que el primero se puede ejecutar en un sistema de computación grid ya que usa el sistema Sparrow (ver Sección 7.4.2). Los datos incluidos están calculados sobre 30 ejecuciones independientes.

En los experimentos realizados, se han utilizado los equipos de siete laboratorios del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga. Todos ellos están equipados con procesadores de última generación Intel Core 2 Duo a 3 GHz. Esto significa que cada procesador tiene dos núcleos y Sparrow asume que existen dos procesadores por máquina. A partir de ahora, no obstante, se utilizará el término procesador aunque la palabra correcta es núcleo. El sistema completo está compuesto por más de 300 procesadores que están interconectados mediante una red Fast-Ethernet a 100 Mbps.

Rendimiento paralelo

El primer estudio que presentamos sobre assNSGA-II se encarga de analizar el comportamiento paralelo del algoritmo. La Tabla 8.10 incluye los mejores valores, los valores medios, $\bar{x}$, y las desviaciones estándar, σ_n, de diversos indicadores de rendimiento.

La primera medida que aparece en la tabla de resultados es el número máximo de esclavos utilizados en las diferentes ejecuciones. Como se puede observar, hemos conseguido tener computando a la vez un máximo de 288 procesadores, y 215 en media. No se ha llegado a usar los más de 300 procesadores que componen nuestro sistema, pero esto es un comportamiento típico en plataformas grid, en el que el número de máquinas disponibles es muy dinámico y están compartidas, normalmente, por diferentes usuarios.

Una de las ventajas de utilizar sistemas de computación grid es que se pueden resolver, en un tiempo razonable, problemas que de otra forma sería inviable abordar. Así, en la Tabla 8.10 se puede observar que el tiempo total de CPU reportado por Sparrow (es decir, el tiempo total de CPU acumulado por todos los procesadores utilizados) es de casi 6 días, mientras que el tiempo real de cómputo de las ejecuciones es de 1,05 horas (más de 132 veces más rápido). Estos valores dejan claro los beneficios de nuestra propuesta.

Si consideramos ahora el rendimiento paralelo, Sparrow nos muestra que el valor de este indicador de rendimiento es de casi el 70 % en media, llegando a un 95 % en el mejor caso, lo que es un resultado muy relevante para una ejecución paralela sobre un sistema de computación grid compuesto por cientos de procesadores. No obstante, aunque el tiempo de evaluación de la instancia Arno 3.1 es alto, transferir por la red la información necesaria para esta evaluación (1 MB aproximadamente) también es costoso. Esto hace que la tasa computación/comunicación no sea

Tabla 8.10: Rendimiento de assNSGA-II en la instancia Arno 3.1.

Medida	Mejor valor	$\bar{x}_{\pm\sigma_n}$
Número de esclavos	288	$215_{\pm 41}$
Tiempo total de CPU (s)	410.748 (4,75 días)	$505.019_{\pm 26648}$ (5,85 días)
Tiempo real de cómputo (s)	2.475 (0,69 horas)	$3.799_{\pm 1463}$ (1,05 horas)
Rendimiento paralelo	95,64 %	$69,22\,\%_{\pm 16,10\,\%}$
Tiempo ssNSGA-II (s)	127.836 (1,48 días)	$167.577_{\pm 30578}$ (1,94 días)
Eficiencia paralela media		20,52 %

Tabla 8.11: Hipervolumen de ssNSGA-II y assNSGA-II para el problema Arno 3.1.

	HV
Algoritmo	$\bar{x}_{\pm\sigma_n}$
ssNSGA-II	$0{,}4766_{\pm 0{,}0095}$
assNSGA-II	$0{,}4801_{\pm 0{,}0081}$
Test estadístico	–

del todo favorable y, por tanto, la eficiencia paralela se vea afectada. Para mejorar esta medida sería necesario incluir algún tipo de computación intensiva en los esclavos, como por ejemplo una búsqueda local, que permita una tasa más beneficiosa. Está claro que aún hay muchas posibilidades para mejorar la eficiencia de assNSGA-II.

Las dos últimas filas de la Tabla 8.10 muestran tanto el tiempo de ssNSGA-II (secuencial) como la eficiencia paralela respecto al tiempo real de cómputo de assNSGA-II. Así, cuando comparamos los tiempos de ambos algoritmos, la eficiencia cae al 20 %. La explicación es la siguiente. El envío de las configuraciones completas de la instancia Arno 3.1 a través de la red supone una carga de comunicaciones tal, que el tiempo de evaluación de dicha instancia es poco relevante. No obstante, en términos prácticos, ssNSGA-II necesita dos días de cómputo para abordar el problema, mientras que assNSGA-II sólo requiere una hora, por lo que los beneficios siguen siendo relevantes. Pero, además, como se mostrará en la siguiente sección, nuestro algoritmo grid no sólo es mucho más rápido, sino también más efectivo, es decir, es capaz de computar soluciones más precisas.

Comportamiento numérico

En esta sección se analiza la eficiencia numérica de assNSGA-II con respecto a su versión secuencial ssNSGA-II, para lo que volvemos a utilizar el indicador de calidad Hipervolumen. La Tabla 8.11 muestra la media, $\bar{x}$, y la desviación típica, σ_n, de HV sobre 30 ejecuciones independientes de ambos algoritmos para la instancia Arno 3.1. La última fila de la tabla también incluye el resultado del test estadístico, que en este caso indica que, a un nivel del 95 %, no se puede concluir que las muestras con los valores de HV sean significativamente diferentes.

Como se puede observar, la extensión grid, assNSGA-II, es capar de computar frentes con mejor valor de HV (más alto) que su versión secuencial, ssNSGA-II, para la instancia Arno 3.1. A pesar de que las diferencias son pequeñas y de que no se puede garantizar que son significativas, un rápido vistazo a los frentes de Pareto resultantes (Figura 8.11) muestra algunas ventajas de la versión grid respecto a la secuencial. Se puede observar en la figura que, efectivamente, ambos algoritmos obtienen soluciones con un nivel similar de convergencia (los frentes están prácticamente solapados), aunque assNSGA-II, no obstante, es capaz de cubrir zonas en las que ssNSGA-II no encuentra soluciones no dominadas (parte superior izquierda del frente). Se trata de configuraciones de red con un elevado número de emplazamientos activados y, por tanto, de alto coste, en las que se da servicio a una mayor cantidad de tráfico pero también provocando mayores interferencias. Esta situación que mostramos en la Figura 8.11 no es un hecho aislado, sino que aparece con mayor o menor grado en todos los frentes obtenidos.

El hecho realmente importante que hay que destacar aquí, sin embargo, es que el nuevo modelo diseñado para aprovechar la potencia de cómputo de los sistemas de computación grid no sólo es capaz de reducir el tiempo de ejecución de varios días a unas horas, como se ha mostrado en la sección anterior, sino que también parece más efectivo numéricamente (aunque no se pueden garantizar estadísticamente con un nivel de confianza del 95 %). La clave de esta mejora está en la asincronía introducida en assNSGA-II (Sección 7.4.2). Esto es, dado que las soluciones se envían a los esclavos para su evaluación, el orden en el que se envían no coincide con el orden en el que

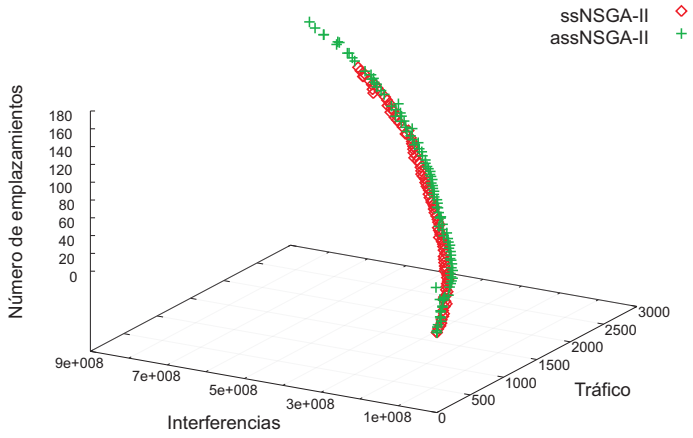


Figura 8.11: Frentes de Pareto de ssNSGA-II y assNSGA-II para la instancia Arno 3.1.

vuelven evaluados y, por tanto, un individuo generado en una iteración t puede llegar de un esclavo en la iteración $t+k$, $t \geq 0$, $k > 0$. Esto provoca una mayor diversidad en la población de assNSGA-II que se traduce en una mejor exploración del espacio de búsqueda y permite cubrir así zonas del frente de Pareto que ssNSGA-II, la versión secuencial, no alcanza. Consideramos, por tanto, que el algoritmo assNSGA-II es una propuesta muy prometedora para resolver el problema ACP.

8.5. Conclusiones

En este capítulos hemos abordado el problema de la planificación de celdas en redes de telefonía (ACP o *Automatic Cell Planning*). Este problema consiste en establecer una configuración de la red de forma que se minimicen costes y se aumente la calidad del servicio. Hemos utilizado tanto una formulación realista del problema como instancias reales, por lo que estamos enfrentando nuestros algoritmos ante un desafío importante. Hemos incluido, también, una revisión de la literatura para ver con qué otras propuestas algorítmicas se ha resuelto este tipo de problemas. En este sentido, las tres metaheurísticas utilizadas, ssNSGA-II, AbYSS y pPAES, son una contribución en el campo, ya que nunca antes habían sido utilizadas.

Una vez introducidas la representación de las soluciones y los operadores básicos que utilizan los diferentes algoritmos, hemos descrito las instancias reales utilizadas y la configuración que utilizan las diferentes técnicas de optimización (probabilidades de recombinación, de cruce, tamaños del conjunto de referencia, etc.). Entonces, hemos incluido tres secciones de resultados.

En la primera, hemos abordado las tres instancias Arno 1.0, Arno 3.0 y Arno 3.1 con nuestras tres propuestas, ssNSGA-II, AbYSS y pPAES (utilizando sólo 1 isla, lo que equivale al algoritmo PAES) y hemos utilizado como base comparativa los dos algoritmos del estado del arte en optimización multiobjetivo: NSGA-II y SPEA2. Los resultados muestran que ssNSGA-II es el que mejor resuelve las tres instancias, superando incluso a los dos algoritmos del estado del arte. Los resultados de AbYSS son novedosos, ya que nunca se había utilizado búsqueda dispersa para resolver este tipo de problemas. Aún así, las diferencias con el resto de algoritmos se mantiene constante y es independiente de la complejidad de la instancia considerada, lo que augura buenas expectativas en los siguientes pasos de esta línea de investigación. PAES, en este caso, es claramente el peor algoritmo, aunque el hecho de no utilizar recombinación de soluciones hacía esperar este resultado ya que su capacidad exploradora queda mermada considerablemente.

El siguiente análisis ha estado dedicado a pPAES. Hemos realizado un estudio del comportamiento paralelo del algoritmo con diferentes configuraciones, así como de su capacidad de búsqueda. En el primer caso, los resultados muestran que pPAES es capaz de obtener importantes reducciones de tiempo cuando se utiliza un número pequeño de procesadores, aunque las ganancias de velocidad decrecen a medida que aumenta el número de máquinas que intervienen en la computación. Éste era un resultado esperable debido a la topología del modelo (anillo unidireccional síncrono). No obstante, el nuevo modelo de búsqueda que se obtiene, si pasamos ahora a evaluar la calidad de las soluciones obtenidas, es más efectivo que el algoritmo PAES. Es decir, según el indicador de calidad Hipervolumen, pPAES es capaz de explorar mejor el espacio de búsqueda de este problema.

Los últimos resultados incluidos hacen referencia a la resolución del problema utilizando sistemas de computación grid. El algoritmo utilizado, assNSGA-II, es la extensión de ssNSGA-II para ejecutar en este tipo de sistemas. Respecto al comportamiento paralelo, assNSGA-II es capaz de reducir el tiempo de cómputo de casi dos días a una hora, utilizando un sistema que cuenta con más de 300 procesadores. Desde el punto de vista de la calidad de las soluciones, la nueva propuesta es, además, más efectiva ya que, utilizando el mismo número de evaluaciones, es capaz de obtener soluciones con un hipervolumen mayor (mejor) que su homólogo secuencial (ssNSGA-II).

Capítulo 9

Resolución del problema de la asignación de frecuencias

Tras estudiar en el capítulo anterior el problema ACP, el siguiente problema que pasamos a resolver es el problema de la asignación de frecuencias o AFP (*Automatic Frequency Planning*). En este caso, se trata de un problema monoobjetivo para el que hemos utilizado las versiones adecuadas de las dos técnicas comunes propuestas, esto es, ssGA y SS, además de un algoritmo de optimización mediante colonias de hormigas (ACO). Su descripción puede encontrarse en el Capítulo 7.

Este capítulo comienza con una revisión de la literatura que muestra los trabajos relacionados con la resolución de este problema usando las tres técnicas propuestas. A continuación, en la Sección 9.2, se detallan la representación y operadores utilizados por estas propuestas algorítmicas. En la siguiente sección se presentan las tres instancias abordadas que se corresponden con tres redes reales de ciudades de EE.UU: Seattle, Denver y Los Angeles. Toda la experimentación realizada se incluye en la Sección 9.4 donde se analizan todos los resultados de ssGA, SS y ACO, así como por el algoritmo GrEA, que aborda el problema utilizando tecnologías grid. El capítulo termina con las principales conclusiones obtenidas.

9.1. Trabajos relacionados

Esta sección se dedica a dar una revisión de trabajos relacionados con la aplicación de algoritmos evolutivos, búsqueda dispersa y colonias de hormigas para la resolución del problema de la asignación de frecuencias. Dado que este problema se planteó a principios de los 70 y que se ha convertido en uno de los problemas clásicos de optimización combinatoria (con muchos bancos de prueba existentes), la producción científica ha sido enorme [1, 82]. Al igual que ocurría con el ACP, la aplicación del algoritmo de búsqueda dispersa es marginal, sólo relacionada con problemas más simples de coloreado de grafos. La sección termina con una revisión de trabajos relacionados con la utilización de colonias de hormigas, nuestra tercera propuesta algorítmica para resolver el problema.

9.1.1. Algoritmos Evolutivos

Como se ha mencionado anteriormente, la utilización de algoritmos evolutivos para la resolución de los distintos problemas que se enmarcan dentro del problema general de la asignación de frecuencias es muy extensa. Una razón fundamental se encuentra en que el problema se convirtió

Tabla 9.1: Algoritmos evolutivos híbridos utilizados para resolver el problema de la asignación de frecuencias.

Ref.	Año	Hibridación con	Versión FAP	Instancias
[124]	1996	Greedy	MI-FAP	Celdas hexagonales (21 celdas)
[4, 5]	2001	Greedy, TS	MI-FAP	Propietarias (5700 TRXs)
[146]	2001	GLS	MO-FAP	CALMA
[240]	2001	Greedy	MS-FAP	Philadelphia + Generadas
[251]	2001	COSEARCH	MS-FAP	Propietarias
[167]	2002	TS	MI-FAP	Propietarias (hasta 639 TRXs)
[171]	2003	Greedy	MI-FAP	Propietarias
[211]	2003	GLS	MS-FAP	Philadelphia
[104, 122]	2004	Markov Decision Process	MI-FAP	Propietarias (5700 TRXs)
[172]	2005	Greedy	MS-FAP	Philadelphia
[223]	2005	Red neuronal	MI-FAP	Philadelphia
[46]	2006	Greedy	MI-FAP	Generadas (hasta 1667 TRXs)
[134]	2007	CAP3	MS-FAP	Philadelphia

muy pronto en un banco de pruebas estándar para la evaluación de nuevas propuestas algorítmicas incluido en la *OR Library* [24, 195]. Su complejidad, así como su aplicación directa en el mundo real, motivaron en gran medida este hecho.

Un primer planteamiento para la revisión de trabajos se podría realizar desde el punto de vista de la versión concreta del problema que estamos resolviendo en esta tesis (ver Capítulo 5): la asignación de frecuencias con mínimas interferencias (MI-FAP) [1]. Trabajos clásicos en la literatura que se incluyen en esta categoría son los de Crisan y Mühlenbein [55, 56], los de Dorne y Hao [73, 74, 107] o los de Hurley *et al.* [57, 58, 120]. Sin embargo, creemos mucho más relevante un análisis según el tipo de algoritmo evolutivo utilizado para resolver cualquiera de las diferentes versiones del problema (todas caracterizadas por una enorme cantidad de restricciones). En este sentido, y debido al elevado número de publicaciones, nos hemos visto forzados a restringirnos sólo a las propuestas en los que los EAs se hibridan de alguna forma con otro método de optimización para resolver mejor el problema [235], que es la aproximación que se sigue en esta tesis. La Tabla 9.1 muestra un resumen de los trabajos más importantes de la literatura. Se han incluido 5 columnas que, respectivamente, indican referencia bibliográfica en cuestión, el año de publicación, el método utilizado para hibridar con el algoritmo evolutivo, la versión del problema que resuelve y las instancias utilizadas junto con su tamaño (cuando esta información está disponible).

El primer aspecto que queremos discutir de la tabla tiene que ver con la columna "Hibridación con". Todos los trabajos analizados, a excepción de [251] (fila 5, COSEARCH), introducen otro método de búsqueda dentro del ciclo de funcionamiento del EA (junto con el cruce y la mutación, o sustituyendo a alguno de ellos). Siguiendo la taxonomía y la gramática propuesta por Talbi en [235] para definir algoritmos híbridos, serían algoritmos *LTH(EA(LS))* ó *Low-level Teamwork Hybrids* en el que el método de búsqueda principal es un EA que lleva incorporado un método de búsqueda local (*LS*) que, en nuestro caso, son algoritmos *ad-hoc* como CAP3, búsqueda tabú, GLS (*Guided Local Search*) [248], Markov Decision Processes y una red neuronal. La gramática definida en [235] también permite añadir tres descriptores adicionales que en nuestro caso serían *(heterogéneo, global, general)*. Las definiciones de estos descriptores se encuentra en [235] para aquellos lectores interesados. La idea es que el EA se dedique a explorar el espacio de búsqueda, mientras que la búsqueda local esté orientada a refinar las soluciones que se van generando durante el ciclo evolutivo. Este esquema es especialmente efectivo para el problema de la asignación de frecuencias, ya que los operadores del EA son generalmente aleatorios y rara vez generan soluciones de calidad

(debido a la violación de múltiples restricciones). Ahí es donde entra la búsqueda local, que trata de resolver, de alguna forma concreta que depende de cada aproximación, las violaciones provocadas por estos operadores. Sin lugar a dudas, la utilización de algoritmos greedy (o de *Hill climbing*) es la que tiene mayor incidencia (presente en 8 de los 15 trabajos analizados), principalmente por su facilidad de uso (escasos parámetros a configurar) y su facilidad para incorporar información sobre el problema concreto que se está resolviendo. COSEARCH [251], que es el único algoritmo analizado que no sigue este esquema, también utiliza una búsqueda local en su funcionamiento, junto con otros dos métodos de búsqueda que cooperan a través de una memoria adaptativa. Se trata de una metaheurística con un modelo de búsqueda paralelo heterogéneo [157].

Respecto a las versiones del problema, se puede observar que están presentes las tres más importantes (MO-FAP, MS-FAP y MI-FAP), mostrando la adecuación de estos algoritmos para resolver el problema. Finalmente, el último comentario de esta sección está dedicado a las instancias que se han abordado así como al tamaño de las mismas. Como se observa en la última columna, las instancias de Philadelphia [82] han sido las más utilizadas. Su tamaño, sin embargo, es pequeño y, además, son modelos muy abstractos del problema real (celdas hexagonales). Las instancias más interesantes por su dimensión y modelado que aparecen en la Tabla 9.1 son propietarias, es decir, no están disponibles, o bien han sido generadas muestreando alguna distribución de probabilidad (por ejemplo, los conjuntos de instancias CALMA y CELAR [82]). En este aspecto, nuestro modelo se acerca mucho más a la realidad de la asignación de frecuencias en GSM, permitiendo enfrentar a los algoritmos a los problemas reales que se encuentran los ingenieros del campo.

9.1.2. Búsqueda dispersa

Al igual que ocurría con este algoritmo para el problema de la planificación de celdas, hasta donde conocemos, no existe ningún trabajo que utilice esta técnica para resolver el problema de la asignación de frecuencias. Así, podemos considerar la aplicación de SS para el AFP como una nueva contribución de esta tesis.

9.1.3. Colonias de hormigas

Para terminar con nuestra revisión de la literatura, esta sección se dedica a analizar los trabajos en los que el problema de la asignación de frecuencias se resuelve mediante algoritmos de optimización basados en colonias de hormigas (ACO), nuestra tercera propuesta para este problema concreto. Hemos de mencionar que, a pesar de lo apropiado de esta metaheurística para abordar el FAP, el algoritmo ACO ha sido poco utilizado, ya que únicamente conocemos tres contribuciones relevantes que se discuten más abajo. De hecho, hasta donde conocemos, nunca antes se ha usado este algoritmo para resolver instancias de gran tamaño como las que aquí se consideran (ver Sección 9.3), por lo que su evaluación en este contexto resulta de gran interés para en campo.

El trabajo de Maniezzo y Carbonaro [168] adapta el algoritmo *ANTS*, originalmente diseñado para el problema de la asignación cuadrática (*Quadratic Assignment Problem* o *QAP*), para resolver la asignación de frecuencias con minimización de las interferencias. Las características fundamentales de esta propuesta es que las hormigas siempre trabajan con soluciones factibles en la fase de construcción. Para la evaluación de ANTS, los autores han utilizado las instancias de Philadelphia, CELAR y CALMA [82]. Dado que la mayoría de estas instancias fueron diseñadas para resolver diferentes versiones de FAP (MO-FAP y MS-FAP), han sido adaptadas utilizando las mejores soluciones conocidas de forma que el valor óptimo de la función objetivo sea cero, es decir, que no haya interferencias en la red.

Montemanni, Smith y Allen [184] también han utilizado el algoritmo ANTS para resolver el problema de la asignación de frecuencias, pero una versión diferente: el MS-FAP (*Minimum-Span FAP*), o minimizar la diferencia entre las frecuencias mínimas y máximas que son utilizadas en la

red. Su propuesta funciona fijando primero el número de frecuencias disponibles a un valor suficientemente grande, y así utilizar posteriormente el algoritmo ANTS para minimizar las interferencias. Una vez que no existen interferencias en la red, se reduce el número de frecuencias (*span*) y se vuelve a ejecutar ANTS.

Acan y Günay han propuesto en [3] un ACO con memoria externa para resolver el banco de pruebas Philadelphia. Esta memoria externa incluye información histórica que se incorpora para guiar la búsqueda de la colonia en la iteración actual. En este caso, lo que se almacena son porciones de tamaño variable de las mejores soluciones de iteraciones anteriores de forma que cuando una hormiga va a construir una solución, extrae una de estas porciones utilizando torneo binario y completa la solución utilizando el esquema general de ACO.

Para finalizar, queremos mencionar los trabajos de Abril y Comellas [2, 47], ya que también utilizan hormigas. No obstante, estas hormigas son en realidad agentes que se mueven por el espacio de soluciones del problema y no siguen el modelo ACO clásico (construcción de soluciones, matriz de feromonas, etc.).

9.2. Representación y operadores utilizados

Dado que los esquemas generales de las metaheurísticas utilizadas para resolver este problema ya se dieron en el Capítulo 7, queda ahora definir la representación empleada para las soluciones que van a manipular (Sección 9.2.1), así como los operadores concretos que aplicarán para realizar la búsqueda. Como ocurría en el caso del problema ACP, ambos elementos son comunes a todos los algoritmos, por lo que vamos a poder comparar cómo se adecúan sus motores de búsqueda al AFP. Debido a las características del problema (alta dimensionalidad y elevado número de restricciones), es prácticamente obligatorio utilizar algún método de búsqueda local que trate de corregir, en la medida de lo posible, las numerosas violaciones de restricciones que producen los operadores estocásticos utilizados. La Sección 9.2.2 incluye el método de búsqueda local aplicado. Las siguientes secciones introducen los operadores utilizados por ssGA, SS y ACO.

9.2.1. Codificación de las soluciones

Como se definió en la Sección 5.2, una solución al problema se obtiene asignando a cada TRX $t_i \in T$ una de las frecuencias de F_i, el conjunto de frecuencias válidas de t_i. Una solución se codifica, por tanto, como un array de valores enteros, p, donde $p(t_i) \in F_i$ es la frecuencia asignada al TRX t_i. Es decir, las soluciones que manipulan los algoritmos son planes de frecuencia de las redes de telefonía móvil que usamos como instancias. Esta codificación, algo más estándar que el esquema jerárquico usado en el problema ACP (Sección 8.2.1), permite utilizar operadores conocidos de la literatura, aunque adaptados a las restricciones que impone el propio problema. La Figura 9.1 muestra gráficamente esta representación.

9.2.2. Búsqueda local

La utilización de algún método de búsqueda local para resolver el problema de la planificación de frecuencias suele ser determinante cuando se utilizan metaheurísticas. El enorme número de restricciones que pueden llegar a ser violadas con los operadores, estocásticos en general, hace que la hibridación con un método de este tipo sea prácticamente obligatoria para obtener resultados competitivos.

El método de búsqueda local que se ha diseñado se puede ver en el Algoritmo 9.1. El funcionamiento es como sigue. En primer lugar, se ordenan los TRXs respecto a su coste componente,

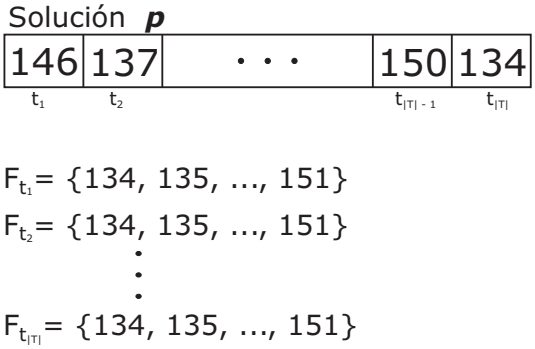


Figura 9.1: Representación utilizada para codificar los planes de frecuencia tentativos.

CC. Dado un plan de frecuencias p y un TRX t, el coste componente de t en p, $CC(p,t)$, se define como

$$CC(p,t) = \sum_{u \in T, u \neq t} C_{\text{sig}}(p,t,u) \tag{9.1}$$

esto es, $CC(p,t)$ es el valor con el que el TRX t contribuye al coste total del plan de frecuencias p (Ecuación (5.2) define C_{sig}). Esta ordenación permite asignar primeramente aquellos TRXs que provocan las mayores interferencias en la red, de forma que podamos obtener soluciones de alta calidad en pocos pasos. Entonces, se recorren todos los TRXs y se elige la frecuencia que más reduce el coste del plan completo (Ecuación 5.1). Dado que para las instancias reales una iteración de este método supondría decenas de miles de evaluaciones de función, en lugar de utilizar la Ecuación 5.1 para calcular el nuevo valor de la función objetivo, hemos diseñado una función incremental que nos permite reducir el tiempo de cómputo del coste de la asignación $p(t_i) = f$. Esta función incremental, Δ, se define como:

$$\Delta(p, p(t_i) = f) = \sum_{t \in \hat{T}_i} (C_{\text{sig}}(p,t,t_i) + C_{\text{sig}}(p,t_i,t)) \tag{9.2}$$

donde $\hat{T}_i = \{t \in T, M(s_{t_i}, s_t) = (\mu, \sigma) \text{ y } \mu > 0\}$, siendo M la matriz de interferencias y s_{t_i} y s_t los sectores en los que están instalados los TRXs t_i y t, respectivamente. Esto es, $\hat{T}_i$ es el conjunto de TRXs que potencialmente pueden interferir con el TRX t_i, según la matriz de interferencias.

Algoritmo 9.1 Pseudocódigo de la búsqueda local para el AFP.

```
1: Entrada: una solución p, un número de pasos d
2: mejorada ← true
3: k ← 1
4: while k ≤ d and mejorada = true do
5:    mejorada ← false
6:    Ordenar cada TRX t_i con CC(p, t_i)
7:    for i ← 1 to n do
8:       Reemplazar la frecuencia p(t_i) con la frecuencia de F_i que más reduce la función de coste
9:       if se ha reducido el coste then mejorada = true
10:      Actualizar CC(p, t_i)
11:   end for
12:   k ← k + 1
13: end while
14: output: una solución p posiblemente mejorada
```

Algoritmo 9.2 Pseudocódigo del operador de mutación

```
1: Entrada: una solución p, una probabilidad p_m
2: for t_i ∈ p do
3:    r ∼ U(0,1)
4:    if r < p_m then
5:       f ← rand(F_i)
6:       p(t_i) ← f
7:    end if
8: end for
```

Finalmente, se actualiza el coste componente de cada TRX y comienza una nueva iteración (línea 10 en el Algoritmo 9.1) de forma que se reasignen primero los TRXs que más contribuyen al coste. El parámetro d permite ajustar el número de iteraciones que se realizan, definiendo así la carga computacional deseada para este método.

9.2.3. Operadores genéticos

El algoritmo ssGA utiliza selección por torneo binario (como ya se indicó en las Sección 7.1.1), además de recombinación uniforme (UX o *Uniform Crossover*) y mutación aleatoria. La recombinación uniforme toma dos padres y genera un único descendiente en el que cada alelo, en este caso la frecuencia de cada TRX, se elige aleatoriamente (con probabilidad uniforme de 0,5) de uno de los padres. Formalmente, dados dos individuos p_1 y p_2, se genera un hijo o de forma que cada TRX t_i toma el valor de uno de los padres con la misma probabilidad:

$$o(t_i) = \begin{cases} p_1(t_i) & \text{si } r < 0{,}5 \\ p_2(t_i) & \text{en otro caso} \end{cases} \tag{9.3}$$

siendo $r \sim U(0,1)$, un valor aleatorio uniforme entre 0 y 1.

El operador de mutación aleatoria, por su parte, modifica la frecuencia asignada a un conjunto aleatorio de TRXs del individuo. Para cada TRX, la reasignación se realiza reemplazando la frecuencia actual por otra cualquiera seleccionada aleatoriamente de entre sus frecuencias válidas. El Algoritmo 9.2 muestra el pseudocódigo de este operador de mutación. Como se puede observar, ambos operadores generan siempre planes de frecuencia válidos de forma que, dado un individuo p, está garantizado que $\neg\exists p(t_i), t_i \in T$, tal que $p(t_i) \notin F_i$.

9.2.4. Métodos de la búsqueda dispersa

En la descripción general de la plantilla de búsqueda dispersa quedaron por definir, debido a su dependencia del problema, los métodos de generación de soluciones diversas, de mejora y de combinación de soluciones. Además, en el método de actualización del conjunto de referencia hay que determinar cómo se mide la distancia entre dos soluciones que, por supuesto, también depende del problema en cuestión. La idea aquí, como ya se ha sugerido, consiste en utilizar los mismos operadores estocásticos procedentes de los algoritmos evolutivos, como primera aproximación a la aplicación de búsqueda dispersa para la planificación de frecuencias.

Generación de soluciones diversas

Para este método hemos utilizado una generación totalmente aleatoria de soluciones. Así, para cada TRX, se le asigna una frecuencia seleccionada de entre sus frecuencias válidas. Hay que tener en cuenta que, dado el tamaño de las instancias (ver Sección 9.3), este tipo de generación de soluciones es lo suficientemente efectiva para cubrir grandes zonas del espacio de búsqueda.

Método de Mejora

La primera idea que surge para definir el método de mejora es la aplicación directa de la búsqueda local descrita en la Sección 9.2.2. No obstante, para este problema en concreto, esto no es suficiente. La exploración queda estancada ya que la búsqueda local obtiene soluciones muy precisas (es decir, mínimos locales de alta calidad) a partir de las cuales la combinación de soluciones no es capaz de generar soluciones aceptables. En el campo de los algoritmos evolutivos está ampliamente estudiado que combinar soluciones (operador de cruce) no funciona como método de recombinación [55, 73] en este problema. Dado que precisamente esa es nuestra aproximación (ver sección siguiente), hemos tenido que incluir más diversidad para poder hacer que la búsqueda dispersa sea capaz de seguir evolucionando. La forma de hacerlo ha consistido en, antes de aplicar la búsqueda local, el individuo sufre un proceso de mutación intensivo utilizando el operador mostrado en la Sección 9.2.3.

Método de combinación de Soluciones

Este método consiste en aplicar el operador de cruce UX descrito anteriormente en la Sección 9.2.3 para combinar, de manera muy rápida, dos asignaciones de frecuencia distintas.

Método de Actualización del Conjunto de Referencia

Este método quedará totalmente definido una vez propongamos la medida de distancia entre dos soluciones que representan dos planificaciones de frecuencia. Sean p_1 y p_2 estas dos soluciones tentativas, entonces se considera que la distancia D entre ellas es:

$$D(p_1, p_2) = \sum_{t_i \in T} d\left(p_1(t_i), p_2(t_i)\right) \tag{9.4}$$

donde $d(f, g) = 1$, si $f \neq g$ y $d(f, g) = 0$, en otro caso. Es decir, se suma 1 por cada valor diferente que exista en ambas planificaciones. Se podrían haber utilizado otras medidas, como por ejemplo, sumar las diferencias entre frecuencias de cada TRX en cada una de las asignaciones. No obstante, algunos experimentos preliminares indican que el comportamiento de la técnica es muy parecido para las diferentes medidas estudiadas.

9.2.5. Métodos de ACO

El esquema particular de ACO utilizado se conoce como $\mathcal{MM}$AS en el *hyper-cube framework* (HCF) [28], cuyo funcionamiento se incluyó en el Capítulo 7. No obstante, dejamos para este capítulo la presentación de los métodos GenerateSolution y LocalSearch, puesto que eran dependientes del problema e iban a compartir elementos con los operadores utilizados por los otros algoritmos. De hecho, el método LocalSearch utilizado para este problema es justamente el la búsqueda local detallada en la Sección 9.2.2. En cuanto al método de generación de soluciones GenerateSolution, que es el último por definir, trabaja como sigue.

La clave de este método es la definición contenida en la matriz de feromonas $\mathcal{T}$, que contiene información probabilística que se utiliza para construir las soluciones. En este trabajo, cada valor $\tau_{ij} > 0$ representa la conveniencia de asignar al TRX t_i la frecuencia $f_j \in F_i$ (es decir, una frecuencia válida de t_i). Así, la solución se construye eligiendo, por cada t_i, una frecuencia de F_i, siguiendo el orden $i = 1, \ldots, n$. Esto se hace como sigue. En primer lugar, se generan las siguientes probabilidades en cada paso de la construcción de la solución:

$$\mathbf{p}(f_{ij}) = \frac{\tau_{ij} \cdot \eta_{ij}}{\sum_{f_{il} \in F_i} \tau_{il} \cdot \eta_{il}} \ , \ \forall \ f_{ij} \in F_i \ . \tag{9.5}$$

Los valores η en esta ecuación se llaman *información heurística* y se definen, generalmente, utilizando pesos de alguna heurística que depende del problema. Hemos considerado tres opciones:

- **Opción 1:** No se utiliza información heurística alguna, esto es, todos los valores η_{ij} valen 1.
- **Opción 2:** La Ecuación 5.2 pone de manifiesto que la función objetivo penaliza la asignación de frecuencias *similares* a TRXs que están instalados en el mismo sector. Por tanto, el objetivo aquí es precisamente evitar este tipo de asignaciones de la siguiente forma. Sea T_i el conjunto de TRXs que están instalados en el sector $s(t_i)$, y sea $\hat{T}_i \subset T_i$ el conjunto de TRXs que ya tienen asignada una frecuencia. Entonces,

$$\eta_{ij} \leftarrow \left(\left(100 \cdot \sum_{t_l \in \hat{T}_i} \delta(p(t_l), f_{ij}) \right) + 1 \right)^{-1} , \tag{9.6}$$

  donde $\delta(x, y) = 1$ si $|x - y| \leq 1$, y $\delta(x, y) = 0$ en otro caso. En esta configuración, la suma de las interferencias se multiplica por 100 para darle más peso respecto a la información de las feromonas. El valor de 100 se ha elegido empíricamente.
- **Opción 3:** Utilizar la función incremental de coste Δ (Ecuación 9.2) para saber el cambio de valor que se produce en la función de coste cuando se realiza la asignación $p(t_i) = f$, $f \in T_i$. Así, el valor de η se define como:

$$\eta_{ij} = ((100 \cdot \Delta(p, p(t_i) = f)) + 1)^{-1} . \tag{9.7}$$

A la hora de escoger una de las posibles frecuencias para el TRX t_i, se procede como sigue. Primero, se genera un número aleatorio $r \in [0, 1]$. En el caso que $r < d_{\text{det}}$, la frecuencia $f \in T_i$ se selecciona de forma que $\mathbf{p}(f) \geq \mathbf{p}(g)$, $\forall g \in F_i$. En otro caso, se elige una frecuencia para t_i mediante una selección por ruleta utilizando las probabilidades definidas en la Ecuación 9.5. Nótese que d_{det} es un parámetro muy importante en el algoritmo ya que determina el porcentaje de pasos de construcción deterministas frente a los probabilísticos.

9.3. Instancias abordadas

Hemos resuelto tres instancias reales del problema de la asignación de frecuencias que se corresponden con tres ciudades de EE.UU., Seattle, Denver y Los Angeles. Las redes están operativas actualmente y dan servicio a más de 500.000 personas las dos primeras y a casi 4 millones la última, por lo que su resolución es de gran interés práctico. La Tabla 9.2 incluye, por cada instancia, el número de TRXs a los que hay que asignar una frecuencia, así como el número total de frecuencias disponibles. Hay que indicar, no obstante, que no todos los TRXs pueden tener asignada cualquier frecuencia de entre las disponibles. Hemos querido incluir este valor para que el lector se haga una idea del nivel de reutilización del espectro de radio que se da en una red real (por ejemplo, tan sólo 69 frecuencias para casi 42.000 TRXs en la instancia de Los Angeles). En las tres últimas filas de la tabla también se definen los valores usados en las constantes que aparecían en la formulación presentada en el Capítulo 5: K, la penalización por la aparición de interferencias co-canal y de canal adyacente en un mismo sector, y los umbrales c_{SH} y c_{ACR}, que determinan, respectivamente, la calidad mínima de las señales para que se consideren aceptables y la capacidad de los TRXs para recibir una señal en presencia de señales no deseadas en un canal adyacente (*adjacent channel rejection*). Las Figuras 9.2, 9.3 y 9.4 muestran la topología de las tres redes, donde cada triángulo representa a un sector donde operan uno o varios TRXs.

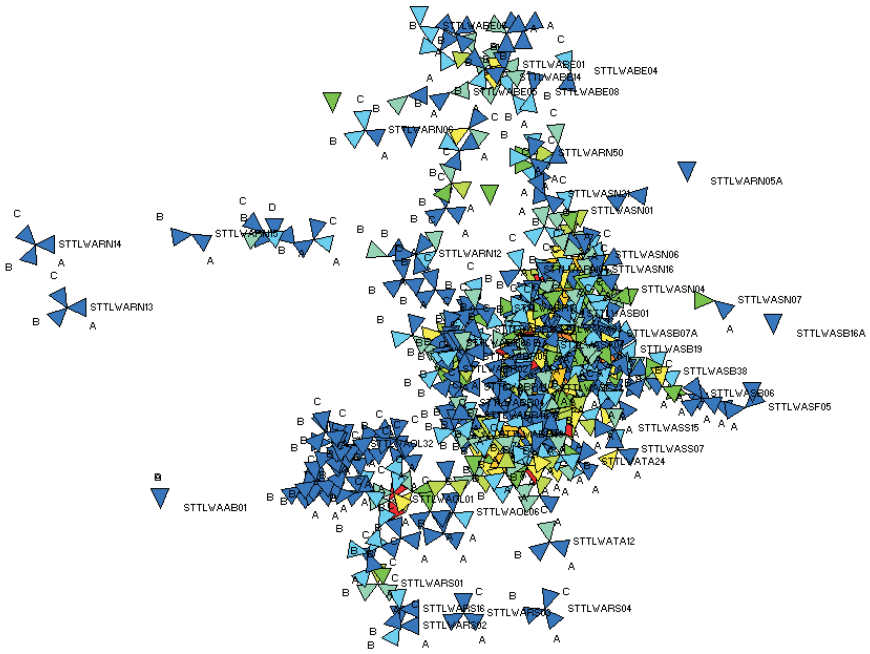


Figura 9.2: Topología de la instancia Seattle.

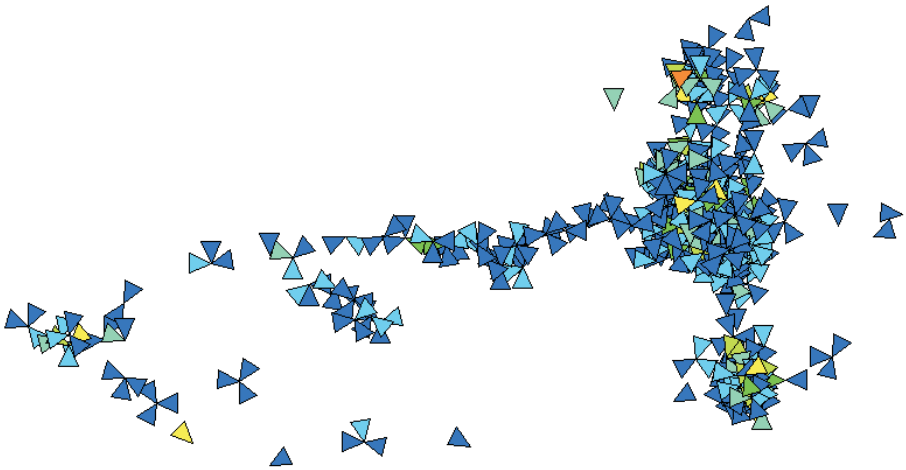


Figura 9.3: Topología de la instancia Denver.

Para terminar esta sección, queremos indicar que los datos utilizados para construir las matrices de interferencia basadas en la distribución de probabilidad del ratio C/I utilizan miles de Informes de Medidas Móviles (MMRs o *Mobile Measurement Reports*) [142] en lugar de modelos de predicción de propagación. Los MMRs son una herramienta mucho más precisa, ya que capturan los patrones de localizaciones de llamada en la red y no dependen de predicciones. Estas propiedades hacen que nuestro problema de asignación de frecuencias en GSM sea más realista que los bancos de prueba estándar que están disponibles [82]. Efectivamente, las instancias que más se parecen a las aquí utilizadas son las del banco de pruebas COST 259, en las que la carga

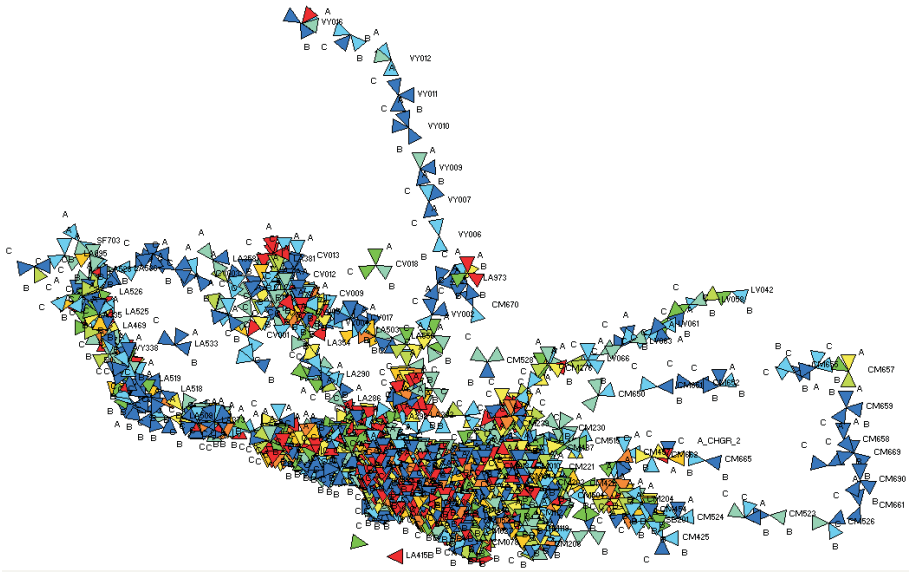


Figura 9.4: Topología de la instancia de Los Ángeles.

Tabla 9.2: Valores que caracterizan las tres instancias resueltas y las constantes utilizadas en la formulación matemática.

Instancia	Seattle	Denver	Los Ángeles
Número de TRXs	970	2.612	41.923
Número de frecuencias diferentes	15	18	69
K		100.000	
c_{SH}		6 dB	
c_{ACR}		18 dB	

de tráfico se genera aleatoriamente de acuerdo a una distribución observada empíricamente. Las señales, además, se predicen utilizando varios modelos de propagación. Las instancias Philadelphia, CELAR y GRAPH [82] son incluso más simples.

9.4. Experimentos

En esta sección se presentan y analizan los resultados obtenidos por las propuestas algorítmicas para resolver el problema de la planificación de frecuencias. Comenzamos detallando la configuración utilizada por ssGA, SS y ACO. En realidad, el ajuste de ACO no es un proceso trivial y hemos decidido incluir el proceso de experimentación previa en la Sección 9.4.2. La Sección 9.4.3 incluye los resultados obtenidos por los tres algoritmos cuando abordan las instancias propuestas. La resolución del problema utilizando sistemas de computación grid se analiza en la última parte de este apartado.

9.4.1. Configuración de los algoritmos

Encontrar la parametrización adecuada de los algoritmos para el problema de la asignación de frecuencias es una tarea crítica, ya que hay que acoplar de forma equilibrada la capacidad explora-

Tabla 9.3: Resultados relativos al uso de información heurística en ACO.

	Sin búsqueda local			Con búsqueda local		
	Mejor	**Media**	**std.**	**Mejor**	**Media**	**std.**
Opción 1	67.460.090,28	69.754.591,73	1.592.508,81	90.515,57	92.021,80	983,03
Opción 2	240.193,31	242.167,29	990,92	89.703,44	91.064,36	887,85
Opción 3	154.835,98	158.307,52	2.140,39	91.496,06	93.494,52	1.511,98

dora de los operadores estocásticos con la enorme características explotadoras de la búsqueda local. Así, incluimos aquí directamente la parametrización de ssGA y SS, y dejamos para la siguiente sección el ajuste de ACO, que resulta mucho menos trivial, debido a los numerosos parámetros que entran en juego.

- **ssGA**. Hemos utilizado una población de 100 individuos, mientras que las probabilidades de cruce y mutación son respectivamente de 1,0 y 0,2. El número de iteraciones de la búsqueda local es 3, es decir, cada invocación de este método supone la realización de tres pasos del mismo.
- **SS**. Para la búsqueda dispersa hemos fijado el tamaño del conjunto inicial P en 20 soluciones. El $RefSet$ tiene tamaño 8, con 4 soluciones para $RefSet_1$ y $RefSet_2$. La combinación de soluciones siempre aplica el cruce uniforme y, en el método de mejora, el operador de mutación se aplica también con probabilidad 0,2. La búsqueda local usa 3 iteraciones.

Teniendo en cuenta que en la resolución de este problema hay involucrados metaheurísticas constructivas como ACO, la postura adoptada aquí para definir la condición de parada ha sido limitar el tiempo de ejecución. Se han considerado 4 límites distintos, 2, 10, 30 y 60 minutos, de forma que se pueda observar el comportamiento de los algoritmos en ejecuciones cortas y largas. Las condiciones experimentales se han mantenido idénticas en todos los casos para garantizar una comparación adecuada: Pentium IV a 2,8 GHz, con la misma carga computational en todas las pruebas.

9.4.2. Ajuste de ACO

En esta sección presentamos un estudio de ajustes de parámetros para el algoritmo ACO. Si bien este proceso de experimentación previa no se ha realizado para el resto de problemas y algoritmos, hemos creído interesante su inclusión puesto que los parámetros a ajustar aquí son menos intuitivos que en el caso del resto de metaheurísticas. En este proceso de ajuste de parámetros, se ha utilizado sólo la instancia de Denver (Sección 9.3), puesto que es la más sensible a los parámetros de búsqueda, según nuestra experiencia.

El primer conjunto de experimentos tienen que ver con las tres opciones propuestas para la información heurística (Sección 9.2.5). Hemos utilizado los siguientes parámetros fijos para ACO: $n_a = 5$, esto es, hay 5 hormigas por iteración (nótese que este parámetro no es crucial y, por tanto, se ha elegido un valor razonable sin realizar experimentos de ajuste) y $r_{\text{det}} = 0{,}5$. Se han aplicado 3 iteraciones en cada invocación de la búsqueda local: $d = 3$. Entonces, hemos ejecutado el algoritmo ACO con las tres funciones de información heurística, además de usando o no búsqueda local. El tiempo de ejecución se ha limitado a 10 minutos, un valor que proporciona ya datos concluyentes. Los resultados, promediados sobre 10 ejecuciones, se muestran en la Tabla 9.3. Por cada configuración, la tabla incluye el mejor valor encontrado, la media y la desviación estándar. Estos resultados nos permiten obtener las siguientes conclusiones. Cuando no se utiliza búsqueda

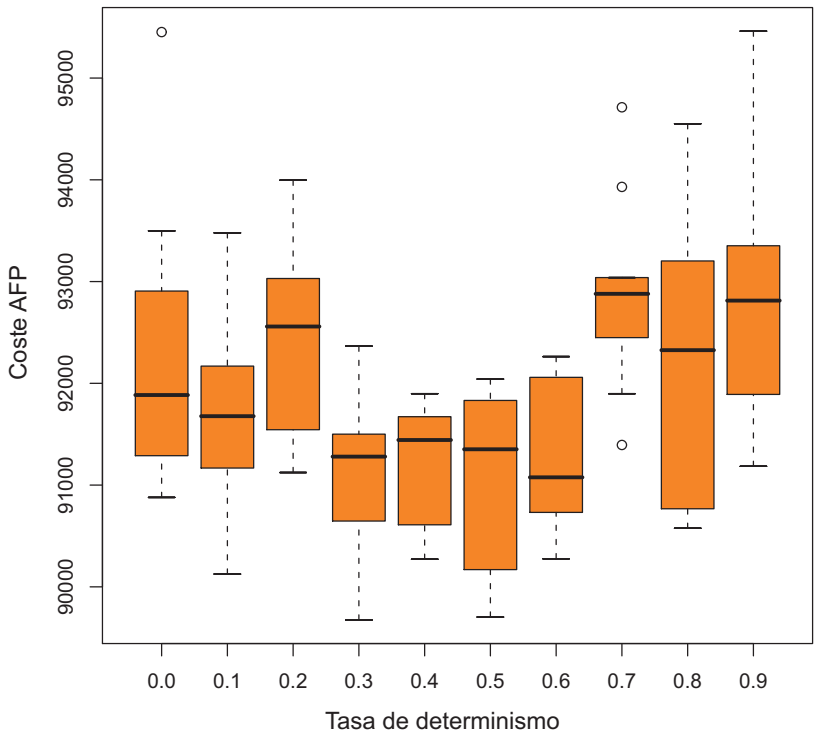


Figura 9.5: Resultados utilizando diferentes valores de la tasa de determinismo (eje x): $r_{\text{det}} \in \{0{,}0\ , 0{,}1\ , \ldots ,\ 0{,}9\}$.

local, la información heurística juega un papel crucial. De hecho, el algoritmo con la Opción 2, respectivamente Opción 3, es sobre 280 veces, respectivamente 430 veces, mejor que la versión que usa la Opción 1 (esto es, sin información heurística alguna). Cuando se compara entre las Opciones 2 y 3, existe una clara ventaja de la última. Una vez que se utiliza la búsqueda local, las tres versiones no difieren mucho entre ellas. De hecho, la mejor configuración es ahora la Opción 2. La razón está basada en el tiempo de cómputo que se necesita para construir una solución. Mientras que los cálculos requeridos por la Opción 2 no provocan un incremento significativo en el tiempo de cómputo, usar la Opción 3 es muy costoso. Más en detalle, la construcción de una solución con la Opción 3 es alrededor de 12 veces más costoso que construir soluciones con las Opciones 1 y 2. Basándonos en estos resultados, nos decidimos por la Opción 2 para todos los experimentos posteriores.

Un segundo conjunto de experimentos ha estado relacionado con el valor del parámetro r_{det}, la llamada *tasa de determinismo*, que establece el porcentaje de pasos de construcción determinista versus probabilística. Se han utilizado 10 valores distintos para este parámetro ($r_{\text{det}} \in \{0{,}0,\ 0{,}1,\ \ldots,\ 0{,}9\}$), manteniendo fijos $n_a = 5$, Opción 2 y 3 pasos de búsqueda local. El límite de tiempo de ejecución también se ha establecido en 10 minutos. Los resultados, que se muestran en la Figura 9.5, indican una clara ventaja de las configuraciones que utilizan valores medios para la tasa de determinismo. Se ha observado también que al comienzo de las fases de reinicio se necesita una mayor tasa de determinismo para alcanzar buenas soluciones rápidamente, por lo que cada una de estas fases comienza con $r_{\text{det}} = 0{,}7$, que después va decreciendo gradualmente hasta llegar a $r_{\text{det}} = 0{,}3$.

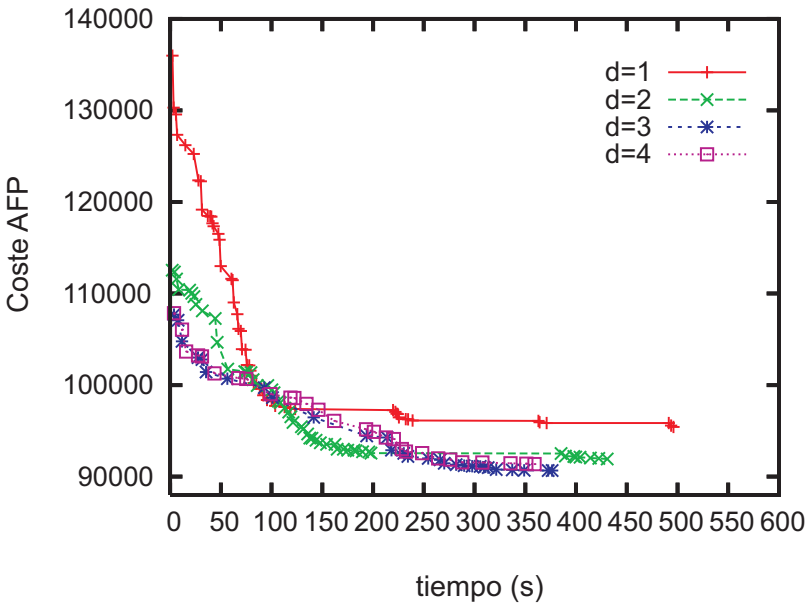


Figura 9.6: Evolution de las mejores soluciones de 4 versiones de ACO con diferentes valores del parámetro d: el número de iteraciones de búsqueda local.

El último conjunto de experimentos realizados para ajustar el algoritmo ACO ha consistido en probar diferentes valores para el número de iteraciones de búsqueda local que se realizan en cada invocación de la misma, esto es, diferentes configuraciones para el parámetro d. Hemos analizado 4 posibilidades: $d \in \{1, 2, 3, 4\}$. La Figura 9.6 muestra la evolución de las mejores soluciones (coste AFP más bajo) a lo largo del tiempo para 4 ejecuciones representativas de las 4 versiones. Se puede observar, en primer lugar, que un único paso de búsqueda local ($d = 1$) es claramente peor. Además, cuando $d \in \{3, 4\}$ el algoritmo exhibe un comportamiento muy similar, e incluso parece algo mejor que cuando $d = 2$. La razón por la que los algoritmos se comporten de forma similar con $d = 3$ y $d = 4$ es el hecho de que, en muchas ocasiones, la búsqueda local no es capar de mejorar más la solución después de 3 pasos. Por tanto, se ha fijado $d = 3$ para la parametrización final de ACO.

9.4.3. Resolución con los algoritmos propuestos

Utilizando las configuraciones anteriores, en esta sección se presentan y discuten los resultados obtenidos por los tres algoritmos para resolver las tres instancias planteadas: Seattle, Denver y Los Angeles. No obstante, dada la importancia de la búsqueda local, en primer lugar hemos incluido un estudio de la efectividad de la misma cuándo se utiliza como resolutor independiente. La intención es demostrar que nuestras propuestas algorítmicas van mucho más allá de las soluciones que puede alcanzar la búsqueda local por sí sola.

Búsqueda local

Para comprobar la eficiencia de la búsqueda local, hemos utilizado una aproximación en la que, partiendo de una solución totalmente aleatoria, el método itera mientras que esta solución se va mejorando, es decir, hasta que ha alcanzado un mínimo local. La Tabla 9.4 muestra la media, $\bar{x}$, y desviación típica, σ_n, del coste final alcanzado, del tiempo que necesitan, y del número de

Tabla 9.4: Resultado de la búsqueda local partiendo de soluciones aleatorias.

	Coste	Tiempo (s)	Iteraciones
Instancia	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$
Seattle	$4.461_{\pm 387}$	$5{,}36_{\pm 1{,}45}$	$8{,}23_{\pm 2{,}01}$
Denver	$107.503_{\pm 2.364}$	$16{,}57_{\pm 5{,}17}$	$9{,}70_{\pm 3{,}01}$
Los Angeles	$1.335.687_{\pm 23.122}$	$32.825{,}83_{\pm 7.775{,}01}$	$23{,}67_{\pm 3{,}51}$

iteraciones del método de búsqueda local, promediando sobre 30 ejecuciones independientes. La plataforma de cómputo es un Pentium IV a 2,8 GHz y los códigos objeto han sido generados todos de la misma forma.

Los valores de los costes obtenidos nos servirán de referencia más adelante para la discusión posterior cuando analicemos los resultados de ssGA, SS y ACO. En este punto nos interesa remarcar los tiempos de cómputo y el número de pasos que necesita el método de búsqueda local para converger, ya que son valores indicativos del coste computacional que requiere. En el caso de Seattle, la instancia más pequeña, este algoritmo converge después de 8 pasos, con un tiempo de 5,36 segundos (0,65 segundos por paso, en media). En la instancia de Denver, con 2612 TRXs, necesita algún paso más para alcanzar un mínimo local, pero con un tiempo medio por paso muy superior: 1,71 segundos/iteración. Sin embargo, cuando abordamos la instancia de Los Angeles (41923 TRXs), el tiempo se dispara. La búsqueda local tarda muchos más pasos en converger, llegando a más de 23 de media, pero, además, cada paso necesita 1386,81 segundos (más de 20 minutos). Aquí no sólo ha intervenido el tamaño de la instancia, sino también el número de frecuencias disponibles por TRXs, que ha pasado de las 15 y 18 en Seattle y Denver respectivamente, a las 69 en esta instancia. De esta forma, si utilizásemos la configuración propuesta para los algoritmos, suponiendo sólo la carga de la búsqueda local, resolver Los Angeles requeriría 4160,43 segundos por cada búsqueda local aplicada (3 pasos), lo que se sale claramente de una escala temporal abordable. Esto nos ha llevado a considerar sólo Seattle y Denver en la posterior evaluación de los tres algoritmos propuestos.

Evaluación de ssGA, SS y ACO

Las Tablas 9.5 y 9.7 incluyen la media, $\bar{x}$, y la desviación típica, σ_n de ssGA, SS y ACO para resolver, respectivamente, las instancias de Seattle y Denver, utilizando 4 límites de tiempo distintos. Los resultados de los test de comparaciones múltiples también se han calculado en este caso y aparecen en las Tablas 9.6 y 9.8.

El primer análisis que incluimos aquí trata de poner en contexto los valores de los costes obtenidos por nuestras tres propuestas algorítmicas para Seattle y Denver, comparándolos con los que ha alcanzado la búsqueda local (Tabla 9.4). En caso de Seattle, ya con 120 segundos como límite de parada, ssGA, SS y especialmente ACO, son capaces de reducir considerablemente el coste de los planes de frecuencia resultantes respecto a los de la búsqueda local (un 43 %, 26 % y 67 %, respectivamente). Para Denver, las reducciones en 120 segundos existen también, llegando a

Tabla 9.5: Resultados de ssGA, SS y ACO para la instancia Seattle utilizando 4 límites de tiempo.

	Seattle			
Límite de tiempo (s)	120	600	1800	3600
Algoritmo	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$
ssGA	$2.550_{\pm 176}$	$1.692_{\pm 92}$	$1.619_{\pm 66}$	$1.573_{\pm 66}$
SS	$3.338_{\pm 201}$	$2.501_{\pm 134}$	$2.112_{\pm 157}$	$1.869_{\pm 166}$
ACO	$1.896_{\pm 132}$	$1.488_{\pm 86}$	$1.361_{\pm 119}$	$1.281_{\pm 114}$

Tabla 9.6: Resultados del test de comparaciones múltiples para la instancia de Seattle.

SS	120	+							
	600		+						
	1800			+					
	3600				+				
ACO	120	+				+			
	600		+				+		
	1800			+				+	
	3600				+				+
	Tiempo	120	600	1800	3600	120	600	1800	3600
		ssGA				SS			

Tabla 9.7: Resultados de ssGA, SS y ACO para la instancia Denver utilizando 4 límites de tiempo.

	Denver			
Límite de tiempo (s)	120	600	1800	3600
Algoritmo	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$	$\bar{x}_{\pm\sigma_n}$
ssGA	$105149_{\pm 2532}$	$91610_{\pm 1474}$	$89214_{\pm 848}$	$88463_{\pm 836}$
SS	$106625_{\pm 1605}$	$98356_{\pm 1118}$	$95016_{\pm 984}$	$94004_{\pm 1107}$
ACO	$94439_{\pm 1363}$	$91984_{\pm 967}$	$90862_{\pm 887}$	$90072_{\pm 743}$

Tabla 9.8: Resultados del test de comparaciones múltiples para la instancia de Denver.

SS	120	+							
	600		+						
	1800			+					
	3600				+				
ACO	120	+				+			
	600		−				+		
	1800			+				+	
	3600				+				+
	Tiempo	120	600	1800	3600	120	600	1800	3600
		ssGA				SS			

un 84 % en el caso de ACO. Tenemos entonces que, ya desde las fases iniciales de las ejecuciones, las tres metaheurísticas alcanzan asignaciones de frecuencias que reducen las interferencias existentes en la red, lo que prueba que son capaces de incorporar a su motor de búsqueda la información de las soluciones procedentes del método de búsqueda local que, no lo olvidemos, se utiliza como operador en todas ellas.

Pasemos ahora a comparar las tres metaheurísticas propuestas entre sí. La primera conclusión clara es que tanto ssGA como ACO computan mejores soluciones que SS para las dos instancias consideradas para todos los límites de tiempo fijados (con significancia estadística, como se puede ver en las Tablas 9.6 y 9.8). Este resultado era el esperado, puesto que la utilización de SS para abordar este problema es muy novedosa y podemos considerar este estudio como preliminar. No obstante, la evolución de los costes en las dos instancias consideradas, Seattle y Denver, con los diferentes límites de tiempo (ver Figuras 9.7 y 9.8, respectivamente) nos permite ser optimistas ya que, en ambos casos, la búsqueda de SS no se queda estancada y es capaz de alcazar mejores planes de frecuencia (de menor coste) a medida que incrementa su tiempo de ejecución.

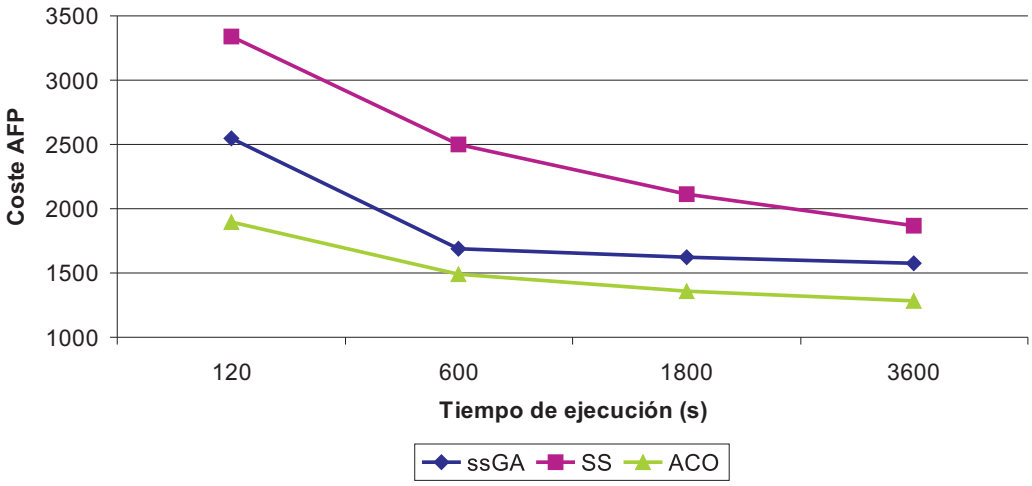


Figura 9.7: Evolución del coste en los 4 límites de tiempo de ssGA, SS y ACO para Seattle.

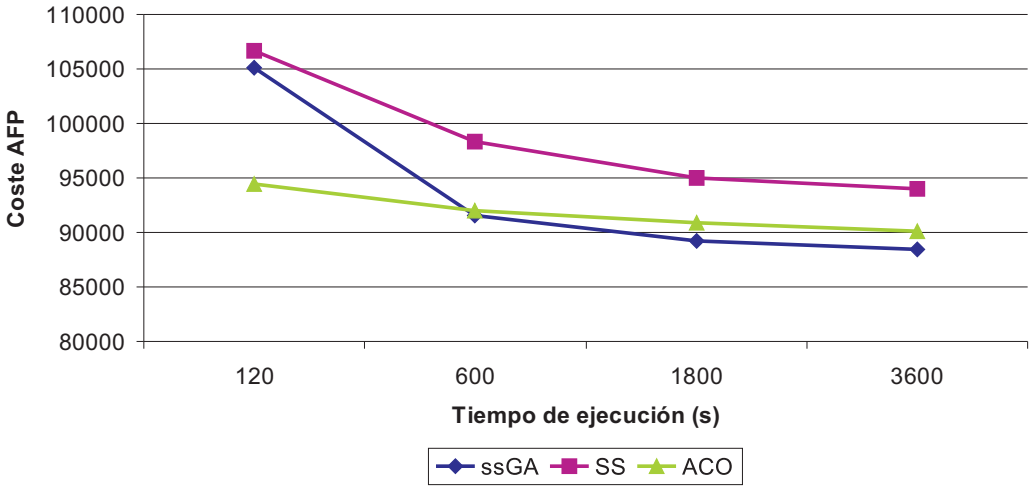


Figura 9.8: Evolución del coste en los 4 límites de tiempo de ssGA, SS y ACO para Denver.

La siguiente conclusión clara que se puede obtener de ambas tablas es que ACO es el mejor algoritmo en las ejecuciones cortas: siempre obtiene los planes de frecuencia con coste más bajo (mejores) para las dos instancias cuando el límite está en 120 segundos. Además, en el caso de Seattle, ACO es claramente el mejor algoritmo para cualquier condición de parada. Las diferencias en los costes de las soluciones de ssGA son notables al principio, con una reducción del 26 % del coste, aunque se mantienen constantes en torno al 15 % en las tres ejecuciones más largas.

No obstante, cuando crece el tamaño de la instancia, es decir, se considera Denver, ACO se ve superado por ssGA a partir de los 600 segundos en adelante (ver Figura 9.8). Las dificultades de los algoritmos basados en colonias de hormigas para instancias grandes del problema de la asignación de frecuencias ya se ha estudiado en la literatura [3] y estos resultados vienen a confirmarlo. La capacidad exploradora de nuestro ssGA, junto con la precisión de la búsqueda local, es lo que le permite seguir descubriendo regiones prometedoras del espacio de búsqueda y el posterior refinamiento de las soluciones.

Como conclusión, podemos decir que ACO supera a ssGA y SS siempre para tiempos de ejecución cortos. Desde el punto de vista de una operadora de telefonía, esta información es muy relevante, ya que le permite disponer de herramientas de optimización que obtienen soluciones de bajo coste en muy poco tiempo. Sin embargo, cuando el tamaño de la instancia crece y hay cierto margen para resolver el problema, el algoritmo seleccionado sería ssGA. Las posibilidades de mejora de SS lo hacen también un algoritmo muy prometedor, puesto que ya ha demostrado su eficacia para otros problemas clásicos de la literatura.

9.4.4. Resultados en sistemas de computación grid

Para resolver el problema AFP en sistema de computación grid hemos utilizado el algoritmo GrEA (Sección 7.4) que, recordemos, está implementado sobre el sistema Condor/MW. No sólo estamos interesados aquí en evaluar el comportamiento paralelo de GrEA sino también su eficiencia numérica. Esto nos lleva a que, para su evaluación, hayamos considerado exclusivamente la instancia de Denver, ya que hay que comparar con ssGA, la versión secuencial de GrEA y, si bien Seattle es demasiado pequeña para usar un sistema grid, Los Angeles es demasiado grande para plantear cualquier algoritmo secuencial para la comparación. Denver es, por tanto, la que presenta las características más adecuadas para este estudio. Después de varios experimentos preliminares, hemos cambiado ligeramente la configuración de los algoritmos. Esta nueva configuración se incluye en la Tabla 9.9. Los cambios van encaminados a aumentar la precisión de ambos así como aumentar la eficiencia paralela de GrEA. La mayor diferencia está en el número de pasos de la búsqueda local, que ha pasado de 3 a 6, lo que permite incrementar la calidad de las soluciones así como hacer más favorable el ratio comunicación/computación. Por último, aquí no tiene sentido utilizar límites de tiempo como condición de terminación de los algoritmos ya que las plataformas de cómputo son distintas; así, hemos fijado un máximo esfuerzo computacional de 50.000 iteraciones para ambos algoritmos.

Si bien los resultados de GrEA están promediados sobre 30 ejecuciones independientes, nos hemos visto forzados a incluir sólo 10 ejecuciones de ssGA con esta configuración, puesto que, sólo una tarda más de 5 días en un Pentium IV a 2,8 GHz. De hecho, trabajar con problemas reales y sistemas de computación grid es muy duro y normalmente se realizan un número pequeño de ejecuciones [141, 151, 173].

Nuestro sistema de computación grid está compuesto por los equipos de siete laboratorios docentes del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, todos ellos están equipados con procesadores de última generación Intel Core 2 Duo a 3GHz. Condor asume, por tanto, que hay dos procesadores por máquina. El sistema está compuesto por más de 300 procesadores que están interconectados mediante una red Fast-Ethernet a 100 Mbps.

Tabla 9.9: Configuraciones de GrEA y ssGA.

Parámetro	**Valor**
Tamaño de la población	100 indiviuos
Recombinación	Cruce uniforme ($p_c = 1{,}0$)
Mutación	Random ($p_m = 0{,}01$)
Pasos de búsqueda local	6
Condición de parada	50000 iteraciones

Tabla 9.10: Medidas de rendimiento de GrEA.

Medida	Mejor valor	$\bar{x}$	σ_n
Max número de esclavos	282	235	24
Número medio de esclavos activos	164	140	10
Tiempo total de CPU (s)	659149 (7,62 días)	680274 (7,87 días)	8476
Tiempo real de ejecución (s)	4094 (1,14 horas)	4879 (1,36 horas)	429
Rendimiento paralelo de MW	73,44 %	71,79 %	0,72 %
Tiempo de ssGA (s)	463014 (3,36 días)	473259 (5,48 días)	12321
Eficiencia paralela media		41,28 %	

Rendimiento del modelo paralelo

Esta sección está dedicada a la evaluación del comportamiento paralelo de GrEA. La Tabla 9.10 incluye el mejor valor, la media, $\bar{x}$, y la desviación estándar, σ_n, de varias medidas de rendimiento.

Si bien nuestro sistema grid está compuesto por unos 300 procesadores, GrEA sólo ha utilizado un máximo de 282 y 235 en media. Como se comentó con anterioridad, éste es un comportamiento típico en sistemas grid, donde el número de procesadores disponibles es dinámico y se tienen que compartir entre los diferentes usuarios.

Si prestamos atención al número medio de esclavos activos, se puede observar que es, en media, de 140. Esto es consecuencia del comportamiento de GrEA bajo Condor/MW. Cada vez que una aplicación MW tiene tareas para computar, solicita al sistema Condor esclavos disponibles. La forma en la que MW solicita más esclavos a Condor es pidiendo un número predeterminado de éstos. Este número fijo es configurable y lo hemos establecido en 100 esclavos, valor que se corresponde con el número inicial de tareas que se crean para evaluar la población inicial, por lo que se garantiza que, al principio, habrá un esclavo por tarea. Este efecto se puede observar en la Figura 9.9. Se puede ver que, al comienzo de la ejecución, el número de tareas se incrementa hasta 100; después de unos pocos minutos, se solicitan otros 100 esclavos al sistema grid. Finalmente, el algoritmo obtiene el resto de procesadores disponibles. Aunque el número total de procesadores se mantiene prácticamente constante (alrededor de 220 en la ejecución mostrada en la Figura 9.9), el número de los que están activos en media se ve afectado por la fase inicial de GrEA.

Una ventaja de usar sistemas de computación grid consiste en resolver, en una cantidad de tiempo razonable, problemas que de otra forma serían considerados inviables. En la Tabla 9.10 se puede observar que el tiempo total de CPU que facilita Condor (es decir, la suma de los tiempos de cómputo de todos los procesadores) es casi de 8 días, mientras que el tiempo real de cómputo en el sistema actual es de 1,26 horas, más de 139 veces más rápida. Estos valores confirman los beneficios de utilizar esta aproximación.

El rendimiento paralelo del que informa Condor está sobre el 72 %, lo que se puede considerar como un resultado excelente en un sistema grid. Sin embargo, el ratio computación/comunicación se podría mejorar utilizando una búsqueda local más intensiva. Se han realizado experimentos adicionales con 10 y 20 pasos de búsqueda local ($d = 10$ y $d = 20$) y el rendimiento paralelo ha aumentado hasta el 81 % y 90 %, respectivamente. La cuestión aquí es que nuestra estrategia de búsqueda local es tan precisa que alcanza un mínimo local después de cinco o seis iteraciones y, por tanto, los resultados numéricos son similares a los ya incluidos aquí. Queda claro que aún hay posibilidades para incrementar la eficiencia de GrEA.

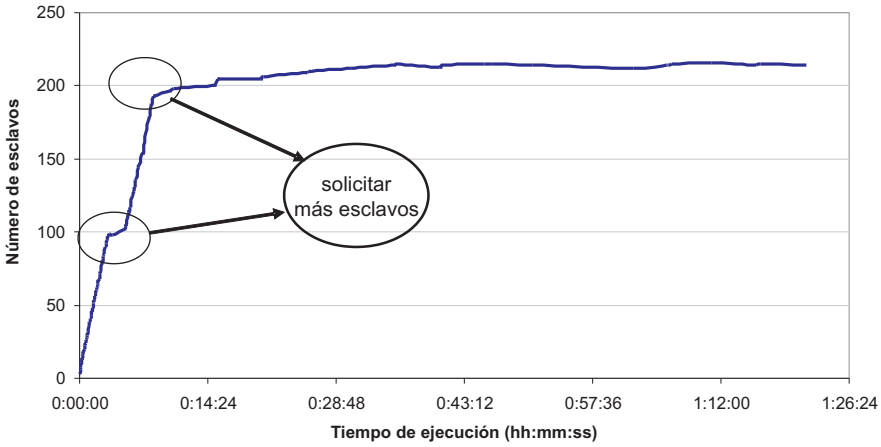


Figura 9.9: Número de esclavos en una ejecución típica de GrEA.

Las dos últimas filas de la Tabla 9.10 incluyen los tiempos de cómputo de ssGA (algoritmo secuencial) así como la eficiencia paralela respecto al tiempo medio de ejecución de GrEA. En este caso, la eficiencia ha bajado hasta el 41,28 %. La explicación es doble. Por un lado, hemos compilado ssGA con diversas opciones de optimización para acelerar la computación todo lo posible, y algunas de estas están relacionadas con el tipo de procesador donde el código se ejecuta; estas opciones no se han utilizado en GrEA debido a la heterogeneidad de procesadores en el sistema grid. Por otro lado, el envío de individuos por la red para su evaluación remota, así como la fase inicial de GrEA donde no se utilizan todas las máquinas disponibles, hacen el ratio comunicación/computación sea menos favorable. De todas formas, en términos prácticos, ssGA requiere más de 5 días para resolver el problema, mientras que GrEA sólo necesita una hora y media, así que los beneficios de nuestra aproximación todavía se mantienen.

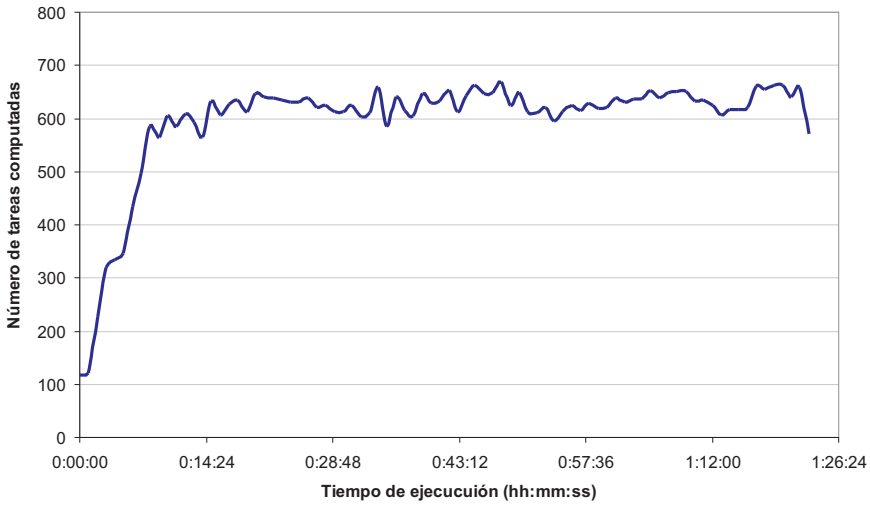


Figura 9.10: Número de tareas por minuto computadas por GrEA.

Tabla 9.11: Resultados de GrEA y ssGA para la instancia Denver.

Algoritmo	Mejor	$\bar{x}_{\sigma_n}$
GrEA	83991,30	$84936,32_{\pm 375,89}$
ssGA	85463,20	$86234,68_{\pm 523,75}$

La productividad, o número de tareas computadas por minutos, suele ser también una medida importante a considerar. Así, en la Figura 9.10 se puede observar que este valor se mantiene prácticamente constante alrededor de las 630 tareas por minuto. Esto no siempre se cumple y en multitud de ocasiones hay que ajustar dinámicamente el grano de la computación remota para que la productividad se mantenga. Esto ocurría con GrEA cuando se aplicó al problema de ensamblaje de fragmentos de ADN en [193].

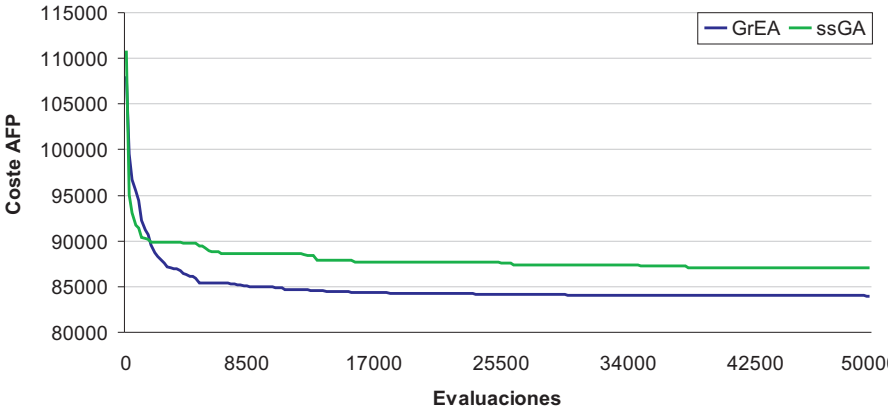
Precisión

En esta sección vamos a mostrar que GrEA no sólo ha sido capaz de aprovechar toda la potencia de cómputo que proporciona nuestro sistema grid, sino que, además, tiene también un mejor comportamiento numérico, es decir, con el mismo esfuerzo computacional es capaz de obtener mejores soluciones. Así, la Tabla 9.11 muestra el coste de la mejor solución, y la media y desviación típica de GrEA y ssGA para la instancia Denver. Los valores son muy indicativos: en media, se ha pasado de 86234,68 a 84936,32, lo que supone un enorme salto de calidad en las soluciones.

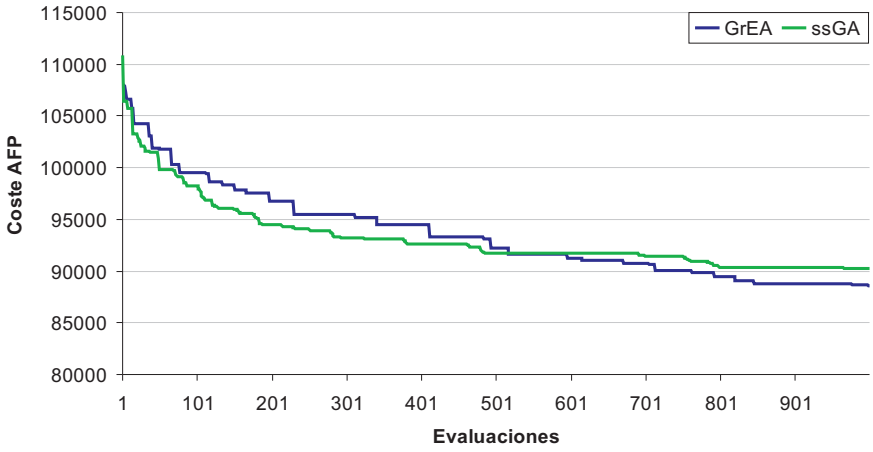
La relevancia de estos resultados tiene que ver con que los dos algoritmos utilizan los mismos operadores y configuración, así como el mismo esfuerzo computacional. La única diferencia es el asincronismo introducido en GrEA, donde los individuos evaluados por procesadores menos potentes son devueltos después de varias iteraciones en los que se han usado máquinas más rápidas para la computación remota. Esto significa la introducción de una mayor diversidad en la búsqueda, especialmente en la fase inicial y que, posteriormente, guía al algoritmo hacia soluciones de más alta calidad [15]. La Figura 9.11 muestra gráficamente este hecho incluyendo la evolución del mejor coste en una ejecución típica de GrEA y ssGA. En la parte (a) de la figura se incluye la evolución del mejor fitness durante la ejecución completa (50000 iteraciones). Se puede observar que GrEA queda estancado más tarde que ssGA y, desde ese punto de estancamiento, ambos son capaces de mejorar la mejor solución sólo ligeramente. En la Figura 9.11b hemos mostrado únicamente las primeras 1000 iteraciones de la misma ejecución para mostrar la mayor diversidad de GrEA al comienzo de la computación. Así, podemos ver que ssGA converge a mayor velocidad que GrEA en las primeras 500 iteraciones, pero encuentra dificultades para continuar mejorando las soluciones después de este punto (regiones planas durante las iteraciones 500 y 700). Mientras tanto, GrEA continúa encontrando soluciones mejores. En el contexto de este trabajo, podemos concluir que desplegar ssGA en un sistema de computación grid, es decir, GrEA, no es sólo una forma de reducir el tiempo de cómputo, sino también de mejorar el modelo subyacente de búsqueda que permite al algoritmo alcanzar mejores soluciones.

9.5. Conclusiones

Resolver el problema de la asignación de frecuencias, esto es, asignar el limitado espectro de frecuencias a los TRXs de una red, es un problema crucial para las operadoras de telefonía móvil. En este capítulo hemos abordado tres instancias reales de este tipo de redes con tres propuestas algorítmicas: un algoritmo genético con selección por estado estacionario, una búsqueda dispersa y una algoritmo de optimización basado en colonias de hormigas. Hemos utilizado también sistemas de computación grid para resolver, de forma más rápida y efectiva, este problema.



(a)



(b)

Figura 9.11: Evolución del coste en GrEA y ssGA durante (a) la ejecución completa y (b) las 1000 primeras iteraciones.

Los resultados muestran que, con las configuraciones utilizadas, el algoritmo ACO es el mejor en las ejecuciones cortas (2 minutos), alcanzando soluciones muy precisas rápidamente. Además, ACO también es el mejor algoritmo en todos los límites de tiempo utilizados como condición de parada para la instancia de Seattle, la más pequeña de las consideradas. No obstante, se ve superado por ssGA cuando consideramos tandas de ejecuciones más largas para la instancia de Denver, de mayor dimensión. En este caso, ssGA ha obtenido siempre los mejores planes de frecuencia ya a partir del límite de 10 minutos. Por último, SS es el que ha reportado los peores resultados para este problema. Dado que es la primera vez que se aplica en este campo, estos resultados se pueden considerar como preliminares y, de hecho, son prometedores en el sentido de que la búsqueda no se queda estancada cuando se deja computar durante más tiempo, además de escalar bien con el problema.

Por último, hemos utilizado el algoritmo GrEA para resolver este problema utilizando como plataforma de cómputo un sistema de computación grid. Los resultados son muy prometedores en dos sentidos. Por una parte, se consigue reducir el tiempo de ejecución de más de 5 días a

poco más de 1 hora y media, lo que supone que GrEA ha sido capaz de aprovechar los recursos computacionales que nos proporciona nuestro sistema grid compuesto por unos 300 procesadores. Por otra parte, GrEA también es más efectivo que su versión secuencial ssGA, es decir, utilizando el mismo esfuerzo numérico, el primero es capaz de encontrar mejores soluciones que el segundo. Esto permite concluir que el nuevo modelo de búsqueda subyacente para desplegar el algoritmo en grid es más efectivo.

Capítulo 10

Resolución del problema de la difusión en MANETs

El último problema que abordamos en esta tesis es el problema de la difusión (o *broadcasting*) en redes ad hoc de dispositivos móviles. Se trata, nuevamente, de un problema multiobjetivo. Las funciones a optimizar son la cobertura alcanzada por los mensajes de broadcasting, el ancho de banda utilizado y la duración de la propia operación de difusión. Hemos propuesto dos formulaciones del problema: *DFCNT*, que utiliza los tres objetivos, y *cDFCNT*, en la que la cobertura pasa a ser una restricción, dejando al problema con los dos objetivos de minimización del ancho de banda y de minimización de la duración. Las propuestas algorítmicas para este problema son, además de las dos metaheurísticas básicas con la que se han resuelto todos los problemas, ssNSGA-II y AbYSS (en el caso multiobjetivo), los algoritmos cMOGA y MOCell, dos algoritmos genéticos celulares para optimización multiobjetivo.

Este capítulo comienza revisando la literatura para identificar las diferentes aproximaciones al problema de la difusión óptima en MANETs. Las Secciones 10.2 y 10.3 describen los operadores específicos utilizados por los algoritmos para este problema así como las instancias que se van a resolver. La Sección 10.4 incluye, en primer lugar, la parametrización empleada por los algoritmos y, después, la presentación y el análisis de los resultados obtenidos. Para terminar, la Sección 10.5 presenta las principales conclusiones que se derivan del estudio realizado en este capítulo.

10.1. Trabajos relacionados

Para este problema no necesitamos subdividir la sección dedicada a la revisión de la literatura, ya que el problema se ha definido durante el desarrollo de esta tesis y, por tanto, estamos creando el cuerpo de conocimiento sobre el mismo. No obstante, hemos identificado una propuesta que utiliza una metodología similar a la nuestra también para un problema en MANETs. Se trata del trabajo de Montana y Redi presentado en [183], en el que se utiliza un algoritmo genético para seleccionar automáticamente los parámetros de un protocolo de encaminamiento para redes ad hoc de forma que este se ajuste a distintos escenarios. El valor de la función objetivo se obtiene mediante el simulador OPNET, donde están implementados tanto el protocolo como los escenarios. Si bien las ideas son similares, nuestro trabajo difiere en varios aspectos. En primer lugar, usamos algoritmos multiobjetivo para resolver el problema, y no una función agregativa como se hace en [183]. Por otra parte, los protocolos que se ajustan son distintos, de difusión en nuestro caso y de encaminamiento en el caso de este trabajo. Finalmente, el enfoque ambos es distinto. Mientras que Montana y Redi tratan de encontrar una configuración del protocolo que valga para diferentes situaciones, nuestro

trabajo consiste en ajustar el protocolo lo máximo posible para un tipo de escenario concreto, intentando garantizar el cálculo de la mejor configuración posible (no se pretende alcanzar un protocolo que funcione bien en cualquier escenario).

10.2. Representación y operadores utilizados

Esta sección incluye la representación de las soluciones y los operadores utilizados por los tres algoritmos planteados en el Capítulo 7 para resolver el problema de la difusión óptima en MANETs: ssNSGA-II, AbYSS y cMOGA/MOCell. Como hemos hecho hasta ahora, continuaremos utilizando los mismos operadores estocásticos procedentes del campo de la computación evolutiva para el método de búsqueda dispersa.

10.2.1. Codificación de las soluciones

En la definición del problema en el Capítulo 6, después de revisar el funcionamiento de DFCN, identificamos 5 parámetros que permiten ajustar el comportamiento del protocolo: *MinGain*, *límiteInferiorDeRAD*, *límiteSuperiorDeRAD*, *proD* y *safeDensity*. Estos cinco parámetros, cuyo conjunto de valores válidos se ha definido en la Sección 6.3, se codifican en un vector de reales (aún siendo algunos valores enteros) que definen el espacio de búsqueda para el problema.

10.2.2. Operadores genéticos

Los operadores de recombinación y mutación que hemos utilizado tanto en ssNSGA-II como en MOCell proceden del campo de la optimización de variables reales. Son el cruce SBX (*Simulated Binary Crossover*) y la mutación polinomial (*Polynomial mutation*) definidas en [66] y utilizados por los algoritmos de referencia en optimización multiobjetivo NSGA-II y SPEA2. SBX simula el comportamiento del cruce de un punto (SPX o *Single Point Crossover*) utilizado en genotipos binarios, mientras que el operador de mutación se caracteriza por usar una probabilidad de mutación polinomial para generar la perturbación que modifica el valor de los genes reales. Ambos operadores se aplican gen a gen con cierta probabilidad p_c y p_m, respectivamente. Además, su comportamiento viene dado también por dos índices de distribución, η_c y η_m, que definen el tamaño de la perturbación aplicado a cada gen.

Tenemos que indicar aquí que existen un par de variables de decisión que son enteras, mientras que los dos operadores trabajan sobre reales. La aproximación seguida aquí ha sido considerar estos valores enteros como reales, de forma que el resultado de las operaciones de recombinación y mutación sobre estas variables se truncan al valor entero más cercano.

10.2.3. Métodos de AbYSS

Los métodos utilizados en AbYSS son los propuestos en la definición original del algoritmo en [192], ya que estamos considerando que el problema tiene variables de decisión reales. Dedicamos una sección para cada uno de ellos.

Método de Generación de Soluciones Diversas

Este método es básicamente el mismo propuesto en [99] y consiste en dividir el rango de cada variable de decisión en un número de subrangos de igual tamaño; entonces, cada nueva solución se obtiene en dos pasos. Primero, se selecciona un subrango de forma aleatoria con una probabilidad inversa y uniformemente proporcional al número de veces que dicho subrango ha sido seleccionado; y, en segundo lugar, se elige aleatoriamente un valor dentro del subrango seleccionado.

Tabla 10.1: Características principales de los entornos propuestos.

		Centro Comercial	**Metropolitano**	**Autopista**
Superficie		40000 m^2	160000 m^2	1000000 m^2
Densidad de círculos		800 (tiendas/km^2)	50 (cruces/km^2)	3 (uniones/km^2)
Dispositivos	Velocidad fuera	0,3–1 m/s	1–25 m/s	30–50 m/s
	Velocidad dentro	0,3–0,8 m/s	0,3–10 m/s	20–30 m/s
	Densidad	2000 disp./km^2	500 disp./km^2	50 disp./km^2
Obstrucción de muros		70 %	90 %	0 %

Método de Mejora

El esquema del método de mejora, ya presentado para el problema ACP (Sección 8.2.3), está basado en un (1+1) EA (*Evolutionary Algorithm*). Éste utiliza un operador de mutación usado como perturbación junto con un test de dominancia de Pareto. El operador de mutación utilizado es la mutación polinomial, usado en tanto en ssNSGA-II como MOCell.

Método de combinación de Soluciones

En la metodología original de búsqueda dispersa, este método computa combinaciones lineales de soluciones del conjunto de referencia. No obstante, nuestra propuesta consiste en usar el operador de cruce SBX presentado anteriormente, ya que hace a AbYSS más robusto [192].

Método de Actualización del Conjunto de Referencia

Lo único que queda por definir de este método es la distancia entre soluciones que, dada la codificación utilizada, es directa: distancia euclídea. Esta medida no se ha incluido en el Capítulo 7 ya que, dependiendo del problema y la representación adoptada, se define de forma diferente. Es decir, no tiene sentido definir distancia euclídea entre las redes del problema ACP.

10.3. Instancias abordadas

Como afirmamos en el Capítulo 6, hemos definido tres entornos distintos para las MANETs simulando tres posibles escenarios que se pueden encontrar en el mundo real: un centro comercial, un área metropolitana y el entorno de una autopista. Dado que consideramos dos posibles formulaciones del problema, *DFCNT* y *cDFCNT*, tenemos tres instancias de cada problema: *DFCNT.CentroComercial* y *cDFCNT.CentroComercial*, *DFCNT.Metropolitano* y *cDFCNT.Metropolitano*, y *DFCNT.Autopista* y *cDFCNT.Autopista*.

Las características principales de cada escenario se explican en esta sección, y se resumen en la Tabla 10.1. Mostramos en la Figura 10.1 un ejemplo de MANET para cada uno de los escenarios estudiados. Estos ejemplos de red se obtuvieron utilizando el interfaz gráfico de Madhoc con la parametrización que se sugirió en nuestro conjunto de problemas propuestos. Consideramos que la difusión se completa cuando la cobertura alcanza el 100 % o cuando no varía en un periodo de tiempo razonable (establecido en 1,5 segundos tras realizar algunos experimentos). Este punto es importante, ya que una condición de parada impropia nos conduciría hacia malos resultados o simulaciones lentas. A continuación se describen dichos entornos.

Entorno de Centro Comercial

El entorno de un centro comercial se propone para emular MANETs en superficies comerciales, en las que se suelen situar tiendas juntas unas con otras en pasillos. La gente va de una tienda a otra

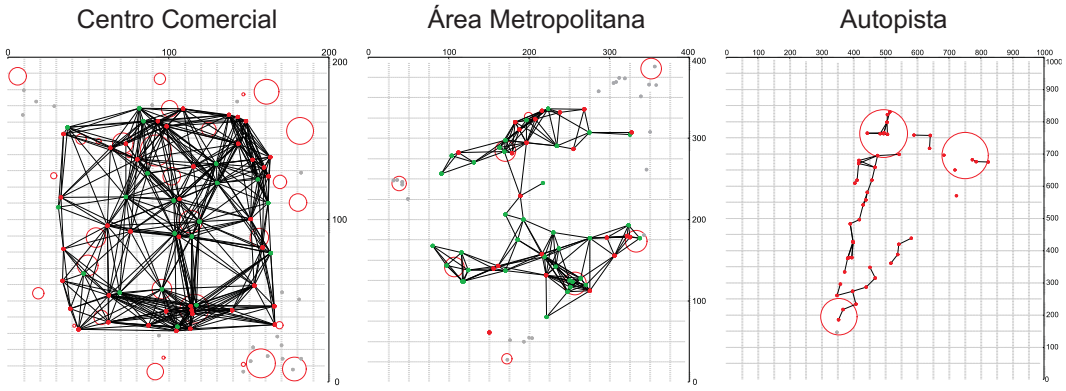


Figura 10.1: Ejemplos de los tres escenarios utilizados.

a través de esos pasillos, parando ocasionalmente para mirar algunos escaparates. Estos centros comerciales suelen tener una elevada concentración de personas (alta densidad de dispositivos), y la gente se comporta de manera distinta (en términos de movilidad) cuando están dentro o fuera de las tiendas. Además, podemos encontrar una alta densidad de tiendas en este tipo de escenarios. Finalmente, en el entorno del centro comercial tanto la movilidad de los dispositivos como la señal de propagación están restringidos por las paredes del edificio.

La parametrización que utilizamos para modelar el entorno del centro comercial en Madhoc es la siguiente. En este tipo de escenarios, la densidad tanto de las tiendas (círculos) como de los dispositivos es normalmente muy grande. Hemos definido para este trabajo un centro comercial de 200×200 metros cuadrados de superficie con densidad de 800 tiendas y 2000 dispositivos por kilómetro cuadrado. Las tiendas son círculos cuyo radio oscila entre 1 y 10 metros, y la obstrucción de las paredes se calcula como una penalización de la fuerza de la señal del 70 %. Finalmente, los dispositivos viajan con una velocidad entre 0,3 y 1 m/s en los pasillos y entre 0,3 y 0,8 m/s cuando están dentro de las tiendas, velocidades muy bajas ya que simulan personas que caminan.

Con respecto al entorno del centro comercial, merece la pena destacar que el grafo de conexión resultante es bastante denso (véase la Figura 10.1). La razón por la que sucede esto es porque la cobertura de los dispositivos móviles se elige aleatoriamente entre 40 y 80 metros, y el área de simulación es pequeño. Por tanto, los problemas *DFCNT.CentroComercial* y *cDFCNT.CentroComercial* son más difíciles de resolver debido al problema de la tormenta en la difusión [194]. Este problema consiste en la aparición de congestiones intensas en la red provocadas por el reenvío de paquetes debido a la alta frecuencia de colisión de paquetes.

Entorno Metropolitano

El segundo escenario realista que proponemos es el entorno metropolitano. Para ello, se sitúan un conjunto de puntos (cruces de calles) en la superficie modelada, que son unidos mediante calles. En este caso, tanto peatones como vehículos se mueven continuamente de un cruce a otro. Como en el mundo real, los dispositivos deben reducir su velocidad cuando se acercan a un cruce.

Para simular este entorno establecemos una superficie de 400×400 metros cuadrados, con una densidad de 50 círculos por kilómetro cuadrado (representando cruces entre las calles) con una superficie circular cuyo radio varía entre 3 y 15 metros. La obstrucción de las paredes en este caso es mayor que en el entorno del centro comercial (llega hasta el 90 %), y la densidad de los dispositivos es de 500 elementos por kilómetro cuadrado. Cuando establecimos la velocidad de los dispositivos tuvimos en cuenta los distintos casos entre personas caminando o viajando en coche,

por lo que su valor oscila entre 0,3 y 10 m/s cuando están en un cruce y entre 1 y 25 m/s en el resto del escenario (cuando están por las calles).

En la Figura 10.1, se puede observar que la red resultante en un área metropolitana no es tan densa como en el entorno del centro comercial. Comúnmente hablando, esta clase de redes se componen de unas pocas subredes, que se conectan normalmente unas a otras con unos pocos enlaces, normalmente uno o dos, o incluso cero (subredes no conectadas). Además, puede que algunos dispositivos no formen parte de ninguna subred (nodos aislados). La topología de esta red puede cambiar muy rápidamente, ya que algunos de los dispositivos viajan a altas velocidades dentro de los vehículos. Todas estas características dificultan la tarea de difusión y hace que para nosotros sea interesante el estudio de esta clase de redes.

Entorno de Autopista

Utilizamos este entorno para simular el comportamiento de las MANETs fuera de las ciudades. Por ejemplo, pensemos en una gran superficie con carreteras y gente viajando en coche. En este contexto, debería haber una baja densidad de dispositivos por kilómetro cuadrado (todos los dispositivos se sitúan en las carreteras), moviéndose todos ellos muy rápido. Además, no existen obstáculos que atenúen la fuerza de la señal en las comunicaciones. Por tanto, utilizamos el escenario *entorno humano* de Madhoc para simular la red con la peculiaridad de establecer una obstrucción de pared del 0 %. La superficie simulada es de 1000×1000 metros cuadrados con una densidad de sólo 50 dispositivos por kilómetro cuadrado. Estos dispositivos viajan a velocidades aleatorias comprendidas entre 30 y 50 m/s. Definimos las carreteras como líneas rectas que conectan dos círculos, y establecemos una densidad de sólo tres círculos (entradas de la autopista y/o salidas). El tamaño de cada círculo (longitud de la entrada/salida) se establece en un valor aleatorio que varía entre 50 y 200 metros (radio de los círculos $\in [25, 100]$ metros).

Se puede ver en la Figura 10.1 cómo la red resultante utilizando esta parametrización está compuesta por un conjunto de múltiples subredes (usualmente inconexas) involucrando un número pequeño de dispositivos (incluso sólo uno). La existencia de estas redes pequeñas y aisladas supone un duro obstáculo para la tarea del protocolo de difusión. Además, la topología de la red cambia muy rápidamente debido a las altas velocidades en el movimiento de los dispositivos. Por tanto, como consecuencia de estas altas velocidades, están continuamente creándose y desapareciendo redes, lo que dificulta aún más el proceso de difusión.

10.4. Experimentación

Esta sección está dedicada a presentar los resultados obtenidos por las propuestas algorítmicas utilizadas para resolver los problemas *DFCNT* y *cDFCNT*. Comenzaremos describiendo los algoritmos de optimización utilizados, así como la configuración de cada uno de ellos. Los primeros resultados presentados son de cMOGA, que fue la primera metaheurística con la que se abordó el problema y que nos ha permitido caracterizar al problema. A continuación se incluyen los resultados de los algoritmos propuestos en esta tesis, ssNSGA-II, AbYSS y MOCell/cMOGA, además de compararlos con los dos optimizadores multiobjetivo del estado del arte, NSGA-II [68] y SPEA2 [262]. La sección termina con los resultados obtenidos al abordar el problema utilizando como plataforma de cómputo un sistema grid.

10.4.1. Configuración de los algoritmos utilizados

En esta sección se incluye la parametrización de todos los algoritmos utilizados para resolver este problema. Para hacer una comparación apropiada, todas las propuestas utilizan la misma

condición de parada: 25.000 evaluaciones de función; y calculan un frente de tamaño máximo 100, es decir, como mucho pueden obtener 100 soluciones no dominadas. Además, debido a la naturaleza estocástica del simulador Madhoc, se realizan cinco simulaciones por cada función de evaluación y se calcula el valor de fitness como la media de los valores obtenidos en cada una de estas simulaciones. Esto tiene una influencia considerable en el tiempo de ejecución para resolver el problema, y explica por qué obtener el número mínimo de ejecuciones independientes (30, al menos) en las pruebas de nuestro algoritmo representa un gran esfuerzo al estudiar el problema.

Partiendo de esta base y de un proceso de experimentación preliminar, la configuración detallada de cada algoritmo utilizado en esta sección es la siguiente:

- **ssNSGA-II**. Nuestra propuesta utiliza una población de 100 individuos. Los operadores de recombinación y mutación (SBX y mutación polinomial) se aplican con probabilidad $p_c = 0,9$ y $p_m = 0,2$, respectivamente. Sus índices de distribución están fijados en $\eta_c = \eta_m = 20$.

- **AbYSS**. Hemos utilizado un tamaño de 20 para la población inicial P y de 10 para $RefSet_1$ y $RefSet_2$. Los valores de índices de distribución para el cruce SBX y la mutación polinomial (métodos de combinación de soluciones y de mejora, que se aplican siempre) son $\eta_c = \eta_m = 20$. El número de iteraciones del método de mejora es 1. Esta es la configuración original propuesta para AbYSS.

- **cMOGA**. Para estructurar la población hemos escogido una rejilla toroidal cuadrada de $10 \times 10 = 100$ individuos. El vecindario utilizado se compone de 5 individuos: el considerado más los que se encuentran situados al norte, sur, este y oeste (vecindario NEWS). Los operadores de selección, recombinación y mutación son, respectivamente, torneo binario, cruce SBX y mutación polinomial. En estos dos últimos operadores, las probabilidades de recombinación y mutación son $p_c = 1,0$ y $p_m = 0,2$, mientras que los índices de distribución utilizados son $\eta_c = 10$ y $\eta_m = 10$.

- **MOCell**. La configuración de la población aquí es la misma que en cMOGA, 100 individuos en una rejilla cuadrada. El vecindario, no obstante, está compuesto por 9 individuos: el actual junto con los ocho que le rodean (COMPACT9). El cruce SBX y la mutación polinomial se aplican con probabilidad $p_c = 0,9$ y $p_m = 0,2$ e índices de distribución $\eta_c = \eta_m = 20$.

- **NSGA-II**. Las probabilidades de cruce y mutación para SBX y mutación polinomial son, como sugieren los autores de NSGA-II [68], 0,9 y 0,2, respectivamente. El valor para los índices de distribución de ambos operadores es 10.

- **SPEA2**. Este algoritmo, que también utiliza los operadores SBX y mutación polinomial [262], ha sido configurado con una población de 100 individuos, además de $\eta_c = 10$ y $p_c = 1,0$ para SBX y $\eta_m = 20$ y $p_m = 0,01$ en la mutación polinomial.

10.4.2. cMOGA: primera aproximación a la resolución del problema

Los primeros resultados que se presentan han sido obtenidos con cMOGA [9, 11], el primer algoritmo utilizado para abordar el problema. En este estudio preliminar hemos analizado los tiempos de cómputo necesarios para obtener una solución al problema y el número de soluciones no dominadas que encuentra cMOGA, para medir el esfuerzo computacional y la complejidad del espacio de búsqueda de los problemas *DFCNT* y *cDFCNT*. Los resultados se discuten en las siguientes secciones.

Tabla 10.2: Resultados de cMOGA para las tres instancias *DFCNT*.

Entorno	**Tiempo (h)**	**Soluciones no dominadas**
DFCNT.CentroComercial	$66{,}12_{\pm 7{,}94}$	$97{,}77_{\pm 3{,}20}$
DFCNT.Metropolitano	$108{,}21_{\pm 8{,}41}$	$93{,}40_{\pm 18{,}02}$
DFCNT.Autopista	$47{,}57_{\pm 0{,}42}$	$52{,}27_{\pm 10{,}99}$

Resolución de *DFCNT*

Ahora presentamos y analizamos los resultados obtenidos para el problema *DFCNT* que, recordemos, se compone de cinco variables de decisión y tres funciones objetivo. En la Tabla 10.2 mostramos la media y la desviación estándar del tiempo de ejecución (en horas) y el número de soluciones no dominadas que encuentra cMOGA para las tres instancias de *DFCNT*: *DFCNT.CentroComercial*, *DFCNT.Metropolitano* y *DFCNT.Autopista*. Como se puede observar, el tiempo de cómputo de una ejecución individual es muy largo, ya que varía desde 1,98 días para el escenario de la autopista hasta 4,51 días en el caso de *DFCNT.Metropolitano*. La razón es el alto coste de calcular la función de fitness, ya que lanzamos cinco simulaciones por cada evaluación, y cada ejecución del simulador necesita entre 1 y 4 segundos. En lo referente al número de soluciones encontradas, los resultados obtenidos son altamente satisfactorios en las tres instancias *DFCNT*, ya que el número de diferentes soluciones no dominadas encontradas es 97,77, 93,40, y 52,27 en media (el máximo es 100) para *DFCNT.CentroComercial*, *DFCNT.Metropolitano* y *DFCNT.Autopista*, respectivamente. Por tanto, permitimos al diseñador elegir entre un amplio rango de posibilidades. Nótese que el número de soluciones que componen el frente de Pareto disminuye conforme la densidad de dispositivos de la red es menor (aumentando la superficie y disminuyendo el número de dispositivos). Esto es porque empleamos el mismo criterio para considerar que termina el proceso de difusión en tres entornos muy distintos. Como trabajo futuro, planeamos adaptar este criterio a cada entorno, lo que esperamos nos conducirá a obtener mejores resultados.

Como ejemplo de la diversidad y del amplio conjunto de soluciones que ofrece el optimizador multiobjetivo, representamos gráficamente en la Figura 10.2 un ejemplo de un frente de Pareto obtenido con cMOGA para los tres entornos estudiados. Las mejores soluciones son las que implican (i) alta cobertura, (ii) bajo ancho de banda, y (iii) una corta duración del proceso de difusión, siendo los parámetros más importantes (i) y (ii). De hecho, los óptimos de Pareto en estos frentes que alcanzan una cobertura de más del 95 % necesitan en media 3,77 segundos y 17,25 mensajes para el centro comercial, y 13,78 segundos y 75,71 mensajes (intensa utilización del ancho de banda) en el caso de un área metropolitana. Por contra, en el caso del entorno de la autopista sólo cinco soluciones del frente de Pareto tienen más del 95 % de cobertura (38 para *DFCNT.CentroComercial* y 16 para *DFCNT.Metropolitano*), alcanzando una media de 74,88 mensajes enviados en 13,37 segundos, que son valores parecidos a los del área metropolitana. Finalmente, podemos notar que si la cobertura no fuese una gran restricción en nuestra red, podríamos usar soluciones muy económicas en términos de tiempo y ancho de banda para los dos entornos.

Comparando los tres gráficos, es posible observar que la difusión es más eficiente en el entorno del centro comercial que en los otros dos casos, ya que siempre tarda menos de 8 segundos (duración), transmite menos de 23 mensajes (ancho de banda), y alcanza más del 40 % de los dispositivos (cobertura). En los entornos metropolitano y de autopista, el proceso de difusión tarda más, con un mayor número de mensajes transmitidos y la cobertura en algunos casos es menor que el 10 %. Finalmente, aunque el frente obtenido para el entorno de la autopista tiene valores límite similares a los que se observan en el área metropolitana, la diversidad es mucho menor en el escenario de la autopista.

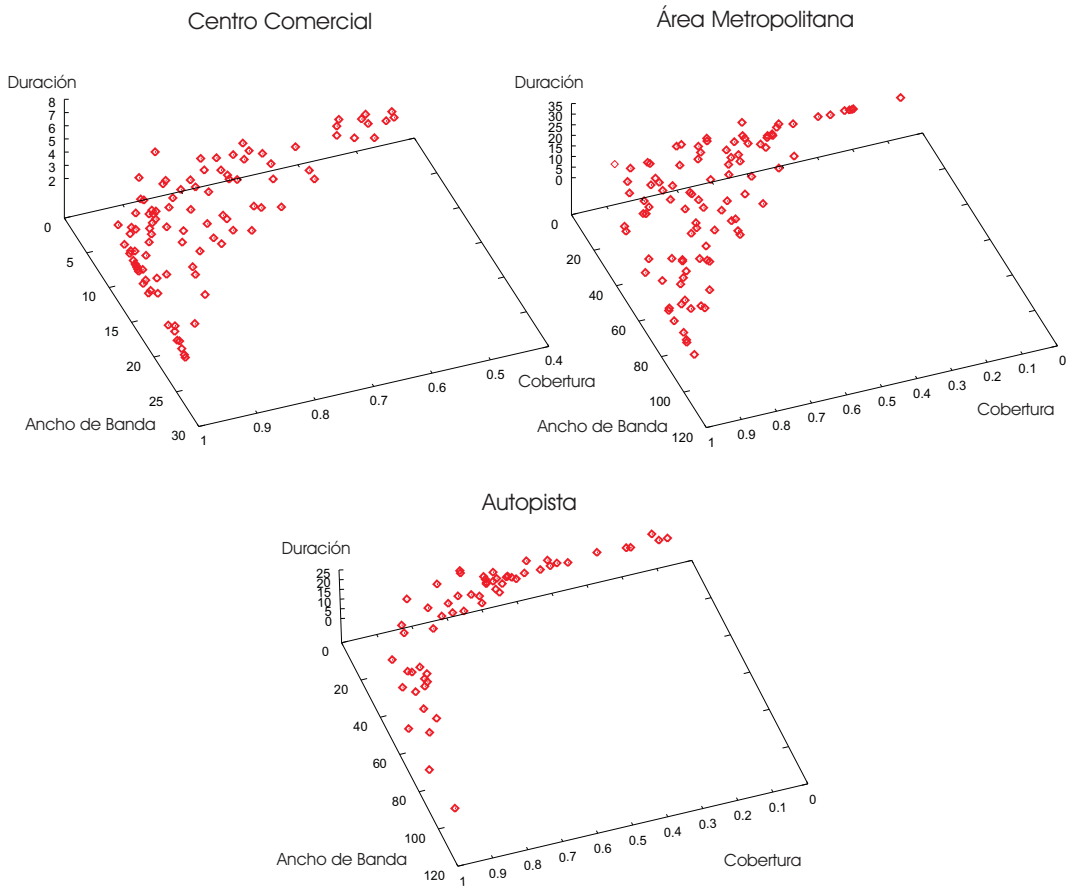


Figura 10.2: Frentes de Pareto para los tres entornos y el problema *DFCNT*.

La diferencia de la cobertura entre los conjuntos de soluciones no dominadas encontrados en los tres entornos es un resultado esperable de forma intuitiva, ya que la probabilidad de tener subredes aisladas (compuestas por uno o más dispositivos) crece conforme aumenta la superficie de simulación y disminuye el número de dispositivos.

Por tanto, la diferencia en la calidad de las soluciones es consecuencia de las diferentes topologías de las redes, ya que el alto grado de conectividad de los dispositivos en el entorno del centro comercial permite que un mensaje llegue a muchos más dispositivos que en el caso de los otros dos entornos estudiados. Contrariamente, este alto grado de conectividad aumenta el riesgo de aparición del problema de la tormenta de difusión, haciendo que *DFCNT.CentroComercial* sea muy difícil de resolver. A partir de nuestros resultados podemos concluir que cMOGA ha tratado el problema con éxito.

Los frentes de Pareto de la Figura 10.2 completan los objetivos de diseño del protocolo de DFCN: la mayoría de los puntos (en el centro de las nubes) se corresponden con conjuntos de parámetros que hacen que DFCN alcance una cobertura cercana al 100 %, manteniendo muy baja la utilización de la red. Lo que hace que el problema DFCNT sea particularmente interesante desde el punto de vista de la aplicación,es que permite al diseñador descartar este comportamiento por defecto estableciendo un *grado de cobertura* para la aplicación de difusión. De hecho, no todas las aplicaciones necesitan maximizar la tasa de cobertura. Por ejemplo, un *anuncio local* –que

Tabla 10.3: Resultados de cMOGA para las tres instancias *cDFCNT*.

Entorno	**Tiempo (h)**	**Soluciones no dominadas**
cDFCNT.CentroComercial	$56{,}53_{\pm 10{,}56}$	$13{,}47_{\pm 2{,}70}$
cDFCNT.Metropolitano	$106{,}15_{\pm 9{,}11}$	$5{,}57_{\pm 1{,}98}$
cDFCNT.Autopista	$46{,}99_{\pm 4{,}32}$	$3{,}40_{\pm 1{,}76}$

consiste en dispersar mensajes publicitarios a dispositivos que estén alejados unos pocos saltos de la fuente– necesita que el proceso de difusión cese después de un rato. A veces también se quiere evitar una alta cobertura. Por ejemplo, intentar conseguir una alta cobertura en una MANET metropolitana (que realmente puede estar formada por miles de dispositivos) es perjudicial, ya que esto nos llevaría a intensas congestiones en la red.

Resolución de *cDFCNT*

Ahora analizaremos los resultados del problema *cDFCNT*, donde la cobertura se ha convertido en una restricción: al menos el 90 % de los dispositivos deben recibir el mensaje de difusión. De esta manera, cMOGA tiene que encontrar soluciones con una relación entre el ancho de banda y la duración del proceso de difusión. Como en el caso de *DFCNT*, la Tabla 10.3 presenta el tiempo medio y el número de óptimos de Pareto que cMOGA es capaz de encontrar en 30 ejecuciones independientes al resolver *cDFCNT* para las tres instancias: *cDFCNT.CentroComercial*, *cDFCNT.Metropolitano*, y *cDFCNT.Autopista*.

En lo referente al tiempo de ejecución, *cDFCNT* se puede resolver un poco más rápido que su equivalente sin restricciones para el entorno del centro comercial (56,53 frente a 66,12 horas), aunque la diferencia es menor (de alrededor del 2 %) en los casos del área metropolitana (106,15 frente a 108,21 horas) y del entorno de la autopista (46,99 frente a 47,57).

Dos razones pueden explicar este comportamiento. Primero, los análisis de dominancia son menos costosos debido a que tienen menos objetivos, reduciendo así el tiempo necesario para comprobar si una nueva solución es dominada o no por el frente actual. Segundo, al encontrar un menor número de puntos para el MOP con restricciones, el análisis anterior es todavía menos costoso debido al bajo número de comparaciones necesarias. Como se afirmó anteriormente, si analizamos las soluciones no dominadas encontradas, los resultados muestran que el número de puntos se ha reducido drásticamente en los tres entornos estudiados respecto a las instancias sin restricciones. Esto indica que la restricción de la cobertura hace que el problema sea muy difícil de resolver (especialmente para los entornos metropolitano y de autopista), quizás excesivamente duro si las soluciones propuestas en *DFCNT* satisfacen al diseñador.

La Figura 10.3 representa tres frentes de Pareto típicos que se corresponden con las tres instancias propuestas de *cDFCNT*. La relación entre las dos funciones objetivo está clara en los tres casos: si un mensaje tiene que ser rápidamente difundido, implica utilizar un gran ancho de banda. Por otro lado, se podrían obtener soluciones baratas en términos de ancho de banda considerando largos tiempos de duración. Una vez más, puede verse que las soluciones para los entornos metropolitano y de autopista son más caras que en el caso del entorno del centro comercial (en lo que se refiere a tiempo y ancho de banda). El bajo número de puntos encontrados para los entornos metropolitano y de autopista con respecto al entorno del centro comercial se puede explicar por la baja cobertura final que se encuentra en las soluciones debido a la topología de la red, como ya comentamos en el caso del problema *DFCNT*.

Como sugerimos antes, todos los protocolos de difusión siguen esta regla: cuanto más oportunistas, más rápidos son (sin considerar el impacto de las colisiones de los paquetes), pero más ancho de banda utilizan. DFCN ha sido diseñado teniendo este aspecto en cuenta: su comporta-

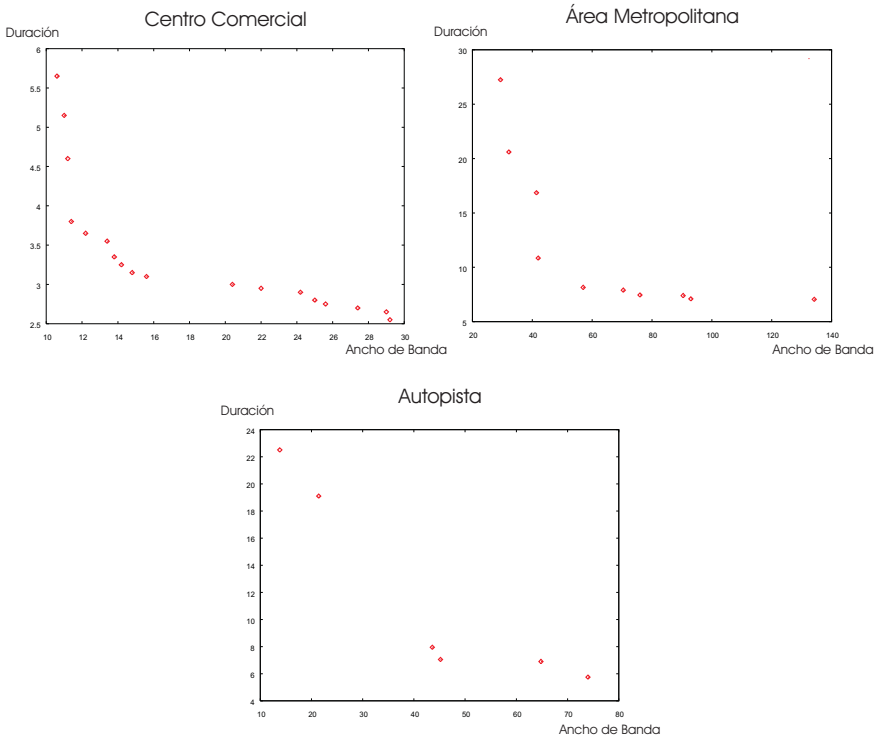


Figura 10.3: Frente de Pareto para los tres entornos y el problema *cDFCNT*.

miento –cuando se usa con los parámetros adecuados– es una excepción a esta regla. De todas maneras, tenemos distintos objetivos en este estudio, ya que estamos buscando el conjunto óptimo de Pareto de los parámetros. Consecuentemente, se muestra en los frentes de Pareto un diverso subconjunto de comportamientos exhibidos por todos los protocolos de difusión. En estos frentes de Pareto podemos observar que conseguir tiempos de propagación muy cortos conlleva un gran ancho de banda y que sólo podemos conseguir un ancho de banda bajo utilizando políticas con pocos reenvíos de paquetes. Aparte de este comportamiento asintótico, los frentes de Pareto también muestran que DFCN puede ser ajustado de tal manera que permita obtener una duración razonablemente corta del proceso de difusión mientras que mantiene una baja ocupación de la red (por ejemplo, número de envíos de paquetes). Al tener garantizada una cobertura elevada ($\geq 90\,\%$), estas parametrizaciones son apropiadas para la mayoría de las aplicaciones de difusión.

10.4.3. Resolución con los algoritmos propuestos

Una vez caracterizados los tres problemas usando cMOGA, en esta sección vamos a abordar el problema de la difusión óptima en MANETs con el resto de metaheurísticas. Dado el elevado coste computacional que supone, hemos elegido la instancia del centro comercial que es la más interesante y la más compleja de resolver. Hemos considerado también la formulación con tres objetivos (*DFCNT*) puesto que la información que proporciona, desde el punto de vista del experto, es mucho más valiosa que cuando se se utiliza *cDFCNT* (dos objetivos más la restricción de cobertura). Hemos usado aquí el Hipervolumen, HV, como indicador de calidad de los frentes resultantes de cada algoritmo, cuya media, $\bar{x}$, y desviación típica, σ_m, sobre 30 ejecuciones independientes se muestra en la Tabla 10.4. El resultado del test de comparaciones múltiples también aparece en la Tabla 10.5.

Tabla 10.4: Hipervolumen alcanzado por los algoritmos para la instancia *DFCNT.CentroComercial*.

	Hipervolumen
Algoritmo	$\bar{x}_{\sigma_n}$
ssNSGA-II	$8,762e-01_{3,7e-03}$
AbYSS	$8,655e-01_{1,3e-02}$
cMOGA	$8,434e-01_{5,6e-02}$
MOCell	$8,752e-01_{2,4e-03}$
NSGA-II	$8,704e-01_{6,1e-03}$
SPEA2	$8,773e-01_{4,5e-03}$
Test estadístico	+

Tabla 10.5: Test de comparaciones múltiples para *DFCNT.CentroComercial*.

AbYSS	+				
cMOGA	+	−			
MOCell	−	+	+		
NSGA-II	+	−	+	+	
SPEA2	−	+	+	−	+
	ssNSGA-II	AbYSS	cMOGA	MOCell	NSGA-II

Comencemos el análisis de resultados considerando nuestras cuatro propuestas para resolver el problema. Según el indicador HV, los frentes obtenidos por ssNSGA-II y MOCell son los de mayor calidad (HV más alto). Si bien este indicador mide tanto convergencia como diversidad, en muchos casos la mejora de estos dos algoritmos respecto a AbYSS y cMOGA se produce en base al primer factor, es decir, proximidad al frente óptimo, lo que supone configuraciones de DFCN que proporcionan operaciones de difusión que alcanzan un mayor número de dispositivos, utilizando para ello menos ancho de banda y también en menos tiempo. Esta situación se refleja en la Figura 10.4, donde se puede ver que las soluciones no dominadas obtenidas por ssNSGA-II dominan a las alcanzadas por cMOGA. El test de comparaciones múltiples (Tabla 10.5) nos muestra, además, que no existen diferencias significativas entre ssNSGA-II y MOCell, pero sí entre estos dos y AbYSS y cMOGA. Este último algoritmo, cMOGA, es el que obtiene los peores (más bajos) valores de HV de entre todas las propuestas. Puesto que se trata de un predecesor de MOCell, que tiene un diseño mucho más avanzado, este resultado era de alguna forma esperado: cMOGA era la primera aproximación en la utilización de un algoritmo genético celular canónico para optimización multiobjetivo.

La Tabla 10.4 también incluye los resultados obtenidos por NSGA-II y SPEA2, los dos algoritmos de referencia en optimización multiobjetivo. Si analizamos ahora la efectividad de nuestras propuestas teniendo en cuenta estos dos algoritmos, se puede observar que SPEA2 alcanza el mayor (mejor) valor para el indicador HV (fondo gris). No obstante, nuestras propuestas ssNSGA-II y MOCell son capaces de computar frentes de Pareto con valores de HV muy similares. De hecho, la Tabla 10.5 indica que las diferencias entre ellos no son estadísticamente significativas (con una confianza del 95 %) y, por tanto, no se puede garantizar que efectivamente SPEA2 sea el mejor algoritmo. Sí podemos afirmar que estos tres algoritmos superan al resto, y con significancia estadística. Tenemos, por tanto, que dos de nuestras propuestas son competitivas con SPEA2 y obtienen mejores valores de HV que NSGA-II, lo que muestra su efectividad y adecuación para resolver el problema de la difusión en MANETs.

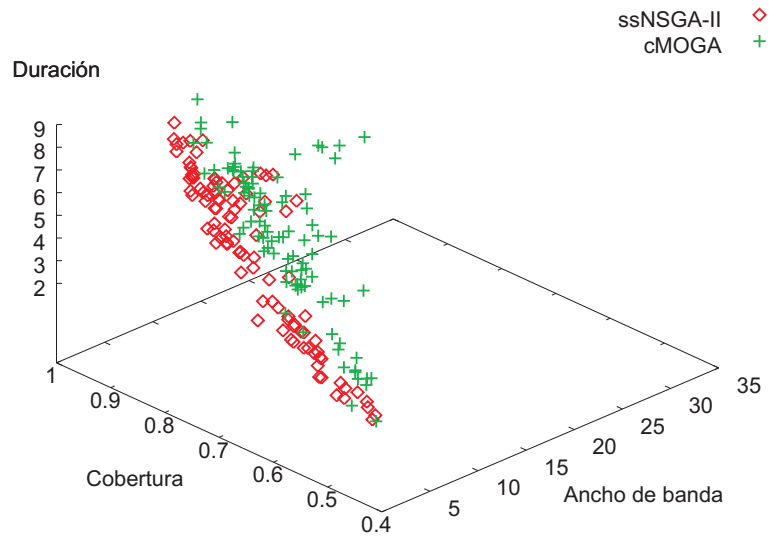


Figura 10.4: ssNSGA-II vs cMOGA para el problema *DFCNT.CentroComercial*.

10.4.4. Resultados en sistemas de computación grid

En esta sección se presentan los resultados de aplicar assNSGA-II (*asynchronous steady state NSGA-II*) nuevamente a la instancia *DFCNT.CentroComercial*, elegida por su complejidad e interés práctico. La parametrización del assNSGA-II es exactamente la misma que ssNSGA-II (Sección 10.4.1), sólo que el primero se puede ejecutar en un sistema de computación grid ya que usa la biblioteca Sparrow (ver Sección 7.4.2). Los datos incluidos son valores medios sobre 30 ejecuciones independientes.

Los experimentos se han realizado el mismo sistema utilizado para resolver el problema ACP: equipos de siete laboratorios del Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, con procesadores de última generación Intel Core 2 Duo a 3GHz. El sistema está compuesto por más de 300 procesadores que están interconectados mediante una red Fast-Ethernet a 100 Mbps.

Los aspectos en los que estamos interesados en estos resultados son dos: la calidad de los frentes obtenidos y la adecuación de la assNSGA-II para aprovechar los recursos computacionales disponibles. Hemos utilizado, de nuevo, el indicador Hipervolumen para el primer caso. La Tabla 10.6 incluye los valores obtenidos por assNSGA-II y la versión secuencial de la que parte, ssNSGA-II. Como se puede observar, los valores de HV para este último son ligeramente mejores (más altos) que los de la extensión grid. La última fila de la tabla indica que el resultado es estadísticamente significativo. A diferencia de lo que ocurría en el problema ACP, donde la normalización realizada a los frentes para calcular de forma fiable el indicador HV hacía que pequeños cambios en este valor

Tabla 10.6: Hipervolumen de ssNSGA-II y assNSGA-II para el problema *DFCNT.CentroComercial*.

	Hipervolumen
Algoritmo	$\bar{x}_{\sigma_n}$
ssNSGA-II	$8,762e-01_{3,7e-03}$
assNSGA-II	$8,703e-01_{3,8e-03}$
Test estadístico	+

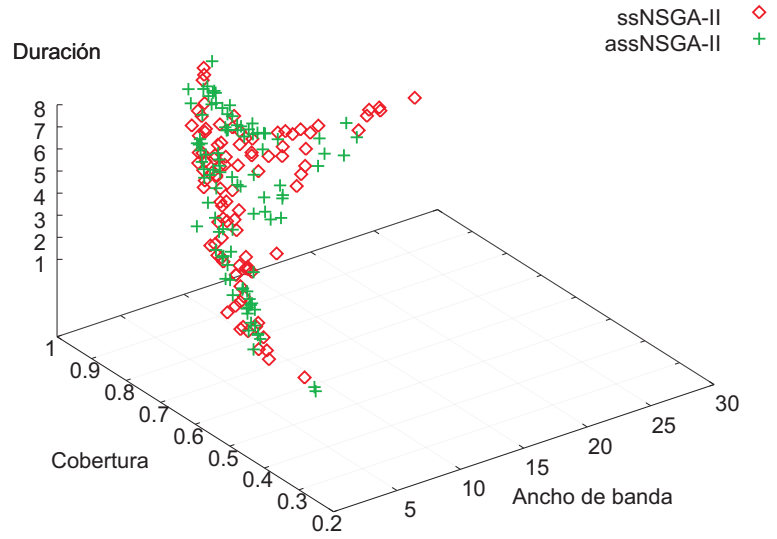


Figura 10.5: ssNSGA-II vs assNSGA-II para el problema *DFCNT.CentroComercial*.

reflejasen diferencias notables en los frentes de Pareto debido a la gran escala de los objetivos (ver Sección 8.4.2), aquí esto no ocurre y las diferencias entre los frentes de ssNSGA-II y assNSGA-II son pequeñas, como se puede observar en la Figura 10.5. Éste es el aspecto de los frentes resultantes de las 30 ejecuciones independientes: en general, se solapan unos con otros excepto en algunas zonas donde ssNSGA-II es ligeramente mejor. En el caso concreto de la Figura 10.5 se puede ver como la versión secuencial alcanza soluciones de máxima cobertura y tiempo mínimo, pero usando un gran ancho de banda (parte superior derecha del frente).

Teniendo en cuenta que assNSGA-II utiliza el mismo esfuerzo computacional (mismo número de evaluaciones de función) que ssNSGA-II, analicemos ahora cómo su rendimiento. Para ello, la Tabla 10.7 incluye los mejores valores, los valores medios, $\bar{x}$, y las desviaciones estándar, σ_n, de diversas medidas de rendimiento.

Como hemos venido haciendo hasta ahora, lo primero que mostramos es el número de procesadores que han intervenido en las ejecuciones. Utilizando de media 214, assNSGA-II ha sido capaz de desplegarse en un máximo de 286. Como era de esperar, tampoco hemos podido utilizar toda la capacidad de cómputo del sistema ya que la disponibilidad de las máquinas es muy variable. No obstante, en media, el tiempo total de CPU consumido por todos los esclavos ha pasado de 1,47 días a tan sólo 21,08 minutos de tiempo de ejecución real (llegando, en el mejor caso, a realizar la ejecución en 7,47 minutos). El rendimiento paralelo reportado por Sparrow es, por tanto, muy relevante: un 83 % de media, alcanzando más de un 99 % en el mejor caso. Este resultado pone de manifiesto la eficiencia de nuestra propuesta.

Tabla 10.7: Rendimiento de assNSGA-II en la instancia *DFCNT.CentroComercial*.

Medida	Mejor valor	$\bar{x}_{\pm\sigma_n}$
Número de esclavos	286	$214_{\pm 84}$
Tiempo total de CPU (s)	118117 (1,37 días)	$126744_{\pm 4735}$ (1,47 días)
Tiempo real de cómputo (s)	448 (7,47 minutos)	$1265_{\pm 1312}$ (21,08 minutos)
Rendimiento paralelo	99,30 %	$83,03\,\%_{\pm 25,19\,\%}$
Tiempo ssNSGA-II (s)	111818 (1,29 días)	$129336_{\pm 43797}$ (1,50 días)
Eficiencia paralela media		47,77 %

Queda ahora evaluar la ganancia respecto al algoritmo secuencial ssNSGA-II. Los datos se muestran en las dos últimas filas de la Tabla 10.7. Se puede observar que aquí la eficiencia paralela ha aumentado hasta casi el 50 % y es que el ratio computación/comunicación es ahora mucho más favorable que en el caso de los problemas ACP y AFP. Efectivamente, enviar a un esclavo un individuo para su evaluación sólo requiere transferir 5 variables reales (los 5 parámetros de DFCN), mientras en que en los casos ACP y AFP había que enviar, respectivamente, la configuración de todos los emplazamientos de la red y las frecuencias asignadas a todos los TRXs (varios cientos de Kbytes). Por tanto, las comunicaciones no suponen un cuello de botella, haciendo que pocos esclavos queden ociosos. En conclusión, assNSGA-II obtiene frentes de calidad similar a los de ssNSGA-II (uno de los más competitivos según vimos en la sección anterior) pero, además, es capaz de reducir el tiempo de cómputo para el problema *DFCNT.CentroComercial* de más de un día a unos 20 minutos. Esto pone de manifiesto las ventajas de nuestra aproximación para resolver problemas del mundo real.

10.5. Conclusiones

Este capítulo está dedicado a la evaluación y análisis de los resultados obtenidos por diferentes metaheurísticas para la resolución del problema de la difusión o broadcasting óptima en MANETs. Este problema es claramente multiobjetivo: maximizar la cobertura de los mensajes de broadcasting, minimizar el ancho de banda utilizado y minimizar la duración del proceso en sí. Para resolver este problema hemos utilizado nuestros algoritmos ssNSGA-II, AbYSS y cMOGA/MOCell. Para demostrar la efectividad de estas propuestas, hemos aplicado también los dos algoritmos de referencia en optimización multiobjetivo: NSGA-II y SPEA2.

El primer estudio que se realiza aquí trata de caracterizar el problema en sí: número de puntos del frente de Pareto que se obtienen, tiempos de ejecución, etc. Para ello utilizamos cMOGA, que fue la metaheurística con la que se abordó por primera vez este problema. Los resultados muestran que, para todas las instancias, los tiempos de ejecución superan claramente el un día de cómputo. El espacio de búsqueda del problema también es muy complejo, ya que cMOGA encuentra muchas dificultades para rellenar completamente un frente de Pareto compuesto por 100 soluciones no dominadas. El problema se presenta, por tanto, como un desafío importante para los algoritmos.

Una vez caracterizado el problema, hemos planteado su resolución de una forma mucho más sistemática con las diferentes propuestas. Para ello sólo hemos utilizado la instancia más interesante y compleja de las tres planteadas, conociendo además que cada ejecución supone entre 1 y 2 días de cómputo. Los resultados aquí muestran que dos de nuestros algoritmos ssNSGA-II y MOCell, junto con SPEA2, han proporcionado los frentes de Pareto de mayor calidad, según el indicador Hipervolumen. Todos superan a NSGA-II, el algoritmo más conocido dentro del dominio multiobjetivo.

Por último, hemos abordado el problema utilizando sistemas de computación grid, para lo que se ha empleado el algoritmo assNSGA-II. En este estudio estamos interesados en dos aspectos: calidad de las soluciones y comportamiento paralelo. En el primer caso, assNSGA-II es igual de competitivo que los mejores algoritmos secuenciales para resolver el problema. Así, el potencial real de esta propuesta viene dado por la habilidad que tiene para aprovechar la enorme capacidad de cómputo que ofrece un sistema grid. De hecho, se ha conseguido reducir el tiempo real de ejecución de más de un día, a tan sólo siete minutos, en el mejor caso, y a 20 minutos, en media. Tanto el rendimiento paralelo de assNSGA-II como la eficiencia paralela obtienen los mejores valores si los comparamos con los dos problemas abordados con anterioridad en esta tesis, ACP y AFP. Esta ventaja radica en un ratio comunicación/computación mucho más favorable, ya que los individuos que se envían para su evaluación remota están compuestos sólo por 5 variables reales, mientras que en los dos casos previos la cantidad de información que hay que enviar es mucho mayor (cercana al MByte en muchos casos).

Parte IV

Conclusiones y Trabajo Futuro

Conclusiones

Esta tesis doctoral ha estado dedicada a la resolución de tres problemas reales en redes de telecomunicaciones utilizando técnicas metaheurísticas. Hay dos aspectos fundamentales de los problemas que han estructurado el desarrollo de este trabajo. Por un lado, dos de los problemas están caracterizados por tener varias funciones que se han de optimizar a la vez, es decir, son problemas multiobjetivo. Por otra parte, en los tres casos hemos abordado no sólo problemas reales, sino instancias también reales. Esto ha supuesto tareas que requieren enormes cantidades de cómputo, incluso cuando se utilizan algoritmos aproximados. Hemos propuesto, por tanto, dos tipos de técnicas avanzadas para enfrentarnos los problemas: técnicas basadas en optimalidad de Pareto, para optimización multiobjetivo, y sistemas de computación grid, para reducir los tiempos de cómputo.

Los tres problemas que se han abordado en esta tesis han sido la planificación de celdas (ACP) y la asignación de frecuencias (AFP), ambos del campo de las redes de telefonía móvil, y el diseño de la estrategia de difusión óptima en redes ad hoc de dispositivos móviles (MANETs), procedente del área de las comunicaciones inalámbricas ad hoc. En el problema ACP, el diseñador de la red ha de seleccionar primero un número de emplazamientos de un conjunto de emplazamientos candidatos donde se instalarán las antenas y, después, configurar los parámetros de propagación de la señal de radio de estas antenas para garantizar una mínima calidad en el servicio, minimizando, a la vez, el coste de desplegar la red. El problema de la asignación de frecuencias, por su parte, consiste en asignar el limitado espectro de frecuencias de radio disponibles a los transmisores/receptores elementales de la red para permitir el establecimiento de comunicaciones, tratando de impedir que se produzca interferencia alguna. Por último, el problema de la difusión óptima en MANETs trata de ajustar una estrategia de difusión para que su rendimiento sea óptimo en un escenario dado. Para cada uno de estos problemas se ha propuesto una definición matemática muy precisa basada conceptos y modelos utilizados en la industria. Se ha dispuesto, además, de datos procedentes de casos reales de los tres problemas, tres instancias por problema concretamente, cuya resolución es de gran interés y suponen un gran desafío para cualquier algoritmo de optimización.

El siguiente paso ha consistido en seleccionar las metaheurísticas con las que vamos a resolver los problemas. Nuestro planteamiento aquí ha sido utilizar dos técnicas comunes con las que abordar los tres problemas: algoritmos evolutivos con selección por estado estacionario, ssGA, y búsqueda dispersa, SS. La primera extensión avanzada que se ha planteado ha estado motivada por el carácter multiobjetivo de dos de los problemas resueltos (ACP y difusión óptima en MANETs). Como resultado hemos diseñado dos nuevos algoritmos, ssNSGA-II y AbYSS. Además, para cada problema, hemos propuesto un algoritmo específico que se adecua a sus características: estrategias evolutivas paralelas para ACP, colonias de hormigas para AFP y algoritmos genéticos celulares para la difusión óptima en MANETs. En total, hemos abordado cada problema con 3 técnicas distintas. Por último, debido al elevado tiempo de cómputo que requiere la resolución de estos problemas, se han propuesto extensiones de los algoritmos ssGA y ssNSGA-II, llamadas GrEA y assNSGA-II, para realizar las computaciones en un sistema de computación grid compuesto por más de 300 procesadores. Mientras que assNSGA-II se ha utilizado para ACP y difusión óptima en MANETs, GrEA se ha aplicado al problema de la asignación de frecuencias (AFP).

En el problema de la planificación de celdas, las propuestas algorítmicas han sido capaces de encontrar soluciones muy eficientes para las instancias consideradas. De hecho, nuestro algoritmo ssNSGA-II ha sido el más efectivo de todos los considerados, superando incluso a los dos algoritmos de referencia en optimización multiobjetivo, NSGA-II y SPEA2. En cuanto a AbYSS, consideramos que la estrategia es muy prometedora, puesto que ha sido la primera vez que se ha aplicado a este tipo de problemas y ha obtenido soluciones competitivas, escalando perfectamente con el tamaño creciente de las instancias abordadas. El algoritmo pPAES ha sido el que ha obtenido configuraciones menos eficientes, pero también ha alcanzado resultados interesantes respecto al nuevo modelo de búsqueda colaborativa que propone ya que ha superado claramente a PAES, el algoritmo en el que está basado. Al abordar este problema utilizando assNSGA-II, la extensión grid de ssNSGA-II, hemos conseguido, no sólo resolver el problema en mucho menos tiempo (de casi dos días a una hora) sino también hacerlo de forma más efectiva, puesto que, utilizando el mismo esfuerzo computacional, ha sido capaz de obtener mejores soluciones.

En la resolución del problema de la asignación de frecuencias, los tres algoritmos propuestos han sido capaces de alcanzar planes de frecuencia que provocan interferencias mínimas en la red. Los resultados han mostrado que el algoritmo basado en colonias de hormigas ha sido el más rápido en encontrar soluciones de calidad y el mejor cuando se consideran instancias más pequeñas. Sin embargo, se ha visto superado por ssGA cuando hemos considerado tandas de ejecuciones más largas para instancias de mayor dimensión. La resolución del problema utilizando tecnologías grid ha resultado un éxito. El algoritmo en cuestión, GrEA, ha sido capar de reducir el tiempo de cómputo de 5 días a poco más de hora y media, consiguiendo, además, obtener mejores planes de frecuencia que su versión secuencial, ssGA, utilizando el mismo esfuerzo numérico.

Para el problema de la difusión óptima en MANETs, nuestras propuestas han resultado, de nuevo, competitivas y, en algún caso, han superado a NSGA-II y SPEA2. Los algoritmos han proporcionado soluciones muy diversas que permiten al diseñador de la red optar por la configuración más adecuada de la estrategia de difusión. La resolución de este problema en nuestro sistema grid con el algoritmo assNSGA-II ha permitido obtener un rendimiento paralelo del 99 % utilizando más de 280 procesadores, lo que ha supuesto un resultado muy relevante. En este caso, las soluciones obtenidas han sido de calidad similar a las del algoritmo secuencial, ssNSGA-II.

Por último, incluimos aquí una valoración global de las aproximaciones algorítmicas propuestas para resolver los tres problemas. Los resultados han revelado que los algoritmos evolutivos con selección por estado estacionario han sido los más adecuados para resolver este tipo de problemas reales que involucran tareas computacionalmente costosas, no sólo en optimización monoobjetivo, sino también en multiobjetivo (ssNSGA-II siempre ha superado a NSGA-II). Además, su funcionamiento ha permitido extenderlos de forma eficiente para poder ejecutarse en sistemas de computación grid, siendo GrEA y assNSGA-II dos claros ejemplos de este hecho. Ha sucedido aquí algo realmente notable y es que GrEA y assNSGA-II no sólo han sido capaces de reducir el tiempo de ejecución de ssGA y ssNSGA-II (sus correspondientes versiones secuenciales) cientos de veces, sino que el modelo de búsqueda subyacente que ha aparece ha sido más efectivo en algunos casos, es decir, ha sido capaz de alcanzar mejores soluciones utilizando el mismo esfuerzo computacional. Respecto al algoritmo de búsqueda dispersa, ha sido la primera vez que se ha aplicado a este tipo de problemas y, por tanto, los resultados han sido ligeramente peores. No obstante, su comportamiento ha sido muy prometedor, puesto que ha escalado bien con el tamaño de los problemas y no se ha estancado durante la búsqueda, por lo que creemos que es capaz de proporcionar soluciones satisfactorias.

Conclusions

This PhD dissertation is devoted to solving three real world problems in telecommunication networks by using metaheuristics. There are two main features of these problems that have guided this work. On the one hand, two of them are engineering problems in which several functions have to be optimized at the same time, i.e., they are multiobjective problems. On the other hand, we have tackled not only real world problems but also real world instances, what involves tasks demanding high computational resources, even when approximate algorithms are used. We have therefore proposed two kind of advances techniques in order to solve these problems: Pareto-based techniques for dealing with multiobjective problems, and grid computing systems for reducing the computing times to affordable values.

The three problems addressed in this thesis are the *Automatic Cell Planning* problem (ACP) and the *Automatic Frequency Planning* problem (AFP), both coming from the cellular networks industry, and the optimal broadcasting design in mobile ad hoc networks (MANETs), belonging to the field of wireless ad hoc networking. In the ACP problem, the network designer has to select first a number of sites from a set of candidate sites where the antennae of the network will be installed and, then, he/she has to configure the propagation features of these antennae so that the radio signal is provided with a minimum quality of services. At the same time, the cost of the network, i.e., the number of installed sites, has to be minimized. The AFP problem lies in assigning the scarce radio spectrum available used for establishing communication links between users to the elementary transceivers of the network, trying to keep interferences as low as possible. Finally, the optimal broadcasting problem in MANETs focusses on tuning a given broadcasting strategy in order to perform the best on a given scenario. For each of these three problems, an accurate mathematical definition has been proposed which is based on concepts and models used in the industry. Additionally, we have realistic data for the three problems, three real world instances concretely, for which their solution is of great interest and it poses a great challenge to any optimization algorithm.

The next step has been to choose the metaheuristic algorithms to be used for solving the problems. Our approach here has been the following. We have selected two techniques which the three problems have been addressed with: steady state evolutionary algorithms, ssGA, and scatter search, SS. The first advanced extension engineered has been mainly driven by the multiobjective nature of two out of the three problems (ACP and optimal broadcasting in MANETs). As a consequence, two new algorithms have been proposed, ssNSGA-II and AbYSS. Additionally, we have developed an especialized algorithm for each problem: parallel Evolution Strategy, called pPAES, for the ACP problem, an ACO (Ant Colony Optimization algorithm) algorithm for the AFP problem, and a cellular GA, named MOCell, for the optimal broadcasting problem in MANETs. Finally, because of the long computation times required to solve these problems, we have proposed an extension of the algorithms ssGA and ssNSGA-II, called GrEA and assNSGA-II respectively, in order to enable them to run on a grid computing system composed of more than 300 processors. Whereas assNSGA-II has been used to solve both ACP and optimal broadcasting in MANETs, GrEA has been applied to the AFP.

For the ACP problem, our proposed metaheuristics has been able to find high quality solutions in the tackled instances. In fact, ssNSGA-II has been the most effective among all the considered algorithms, even outperforming the two reference algorithms in the multiobjective optimization field, NSGA-II and SPEA2. Concerning AbYSS, it can be considered as a very promising strategy because it is the first time that it is applied to this kind of problems and it has reached very competitive solutions, scaling perfectly with the increasing size of the tackled instances. The pPAES algorithm has been clearly the solver performing the worst, but it has reported very interesting results with respect to the newly proposed search model: pPAES has outperformed PAES, the algorithm in which is based on. When addressing this problem with assNSGA-II, the grid extension of ssNSGA-II, we have achieved not only solving the problem in a very much shorter time (from two days to one hour) but also in a more effective way, since the solutions reached were better by using the same computational effort.

When solving the AFP, the three proposed algorithms has been able to compute frequency plans provoking a very low interference level in the network. The results have shown that the ant colony optimization was the fastest metaheuristic in reaching high quality solutions (short term executions) and it was also the best for the smaller instances. However, it was outperformed by ssGA when solving larger instances for longer runs. Solving the AFP problem on grid computing systems has actually succeeded. That is, GrEA has been capable of reducing the computation time from five days to one hour and a half, reaching also better frequency plans than its sequential counterpart, ssGA, and using the same numerical effort.

In the optimal broadcasting problem in MANETS, our algorithmic proposals have reported also very competitive results. They have even outperformed NSGA-II and SPEA2 in several situations. All the algorithms have provided the network designer with a highly diverse set of nondominated solutions which allow them to choose the most appropriate configuration of the broadcasting strategy. Solving this problem on grid computing systems with the assNSGA-II algorithm has reported a parallel efficiency of 99 % using more than 280 processors. In this case, the solutions reached has been of similar quality as those computed by the sequential algorithm, ssNSGA-II.

Finally, we include here a general evaluation of the proposed metaheuristics for solving the three optimization problems coming from the telecommunication industry. The results have revealed that steady state evolutionary algorithms have been the most suitable approach for addressing real world problems involving tasks which demand high computational resources, not only in single objective optimization, but also in the multiobjective case (ssNSGA-II have always outperformed NSGA-II). Moreover, their evolutionary cycle has allow for an efficient extension which has enabled them to run on grid computing systems. GrEA and assNSGA-II are two clear examples. The truly interesting fact here has been that both GrEA and assNSGA-II were not only able of reducing the execution times of ssGA and NSGA-II (their sequential counterparts) hundreds of times, but also the underlying search model which emerged is more accurate, i.e., it reached improved solutions by using the same computational effort. As to the scatter search approach, this is the first time this algorithm has been applied to this kind of problems and, therefore, the reported results were slightly worse. However, it has shown a very promising behavior because it is scalable, i.e., its accuracy did not decrease with larger instances, and, on the other hand, it did not get stuck during long term executions. There is still room for scatter search to provide the tackled optimization problems with satisfactory solutions.

Trabajo futuro

Como fruto de la investigación desarrollada durante esta tesis son muchas las líneas de trabajo futuro que surgen. En los siguientes párrafos sintetizamos las principales.

Planificación de celdas

Creemos que existe todavía mucho margen para abordar la resolución de este problema. Entre las líneas de trabajo que se abren, se incluyen tanto el diseño de nuevos operadores para las metaheurísticas como la evaluación de otras aproximaciones algorítmicas. Dado el espacio de búsqueda tan complejo, sería necesario también incrementar el número de evaluaciones para realizar una mayor exploración y así poder encontrar soluciones más eficientes. Para problemas reales de este tipo, el desarrollo de estrategias de búsqueda local para intensificar la búsqueda en zonas prometedoras también es fundamental. En este sentido, la utilización de tecnologías grid es obligatoria para reducir los tiempos de ejecución a valores aceptables.

Asignación de frecuencias

El modelo matemático desarrollado para modelar este problema es tremendamente preciso y considera conceptos relevantes en la industria de la telefonía móvil. No obstante, se puede extender fácilmente para considerar información más precisa todavía como los tipos de canales a los que se le asignan frecuencia (BCCH – *Broadcast Control CHannel* o TCH – *Traffic CHannel*) y restricciones adicionales que penalizan la asignación de frecuencias similares en TRXs vecinos.

También es de particular interés la formulación multiobjetivo del problema como línea de trabajo para explorar si este enfoque del problema permite encontrar mejores planes de frecuencia, a la vez que se utiliza el mismo esfuerzo computacional. Dado que no existen objetivos adicionales claros que no sean minimizar las interferencias, existen diferentes técnicas en la literatura que ayudan en este sentido, como los la utilización *Helper Objectives*, una técnica novedosa consistente en considerar partes del problema como funciones secundarias que están contrapuestas con el objetivo principal.

Difusión óptima en MANETs

Las oportunidades de investigación que se abren con este problema son muy numerosas. El propio planteamiento del mismo, donde se pueden usar protocolos distintos en diferentes escenarios dependiendo de las necesidades de la red ad hoc, permite proponer una gran cantidad de trabajos nuevos. En particular, hay líneas claras de trabajo en el campo de las VANETs (*Vehicular Ad hoc NETworks*), que actualmente cuentan con un gran auge en el área de las redes ad hoc. A partir de escenarios reales modelados con el simulador más ampliamente utilizado por la comunidad en la actualidad, *ns-2*, y se propone optimizar protocolos de difusión de información a través de esta red.

Metaheurísticas

En el dominio de las metaheurísticas son dos las aportaciones principales de esta tesis, cada una de las cuales abre una posible línea de trabajo futuro. En primer lugar, se han presentado varias nuevas propuestas algorítmicas para optimización multiobjetivo que actualmente forman parte del estado del arte para los conjuntos de problemas de test estándar, como ZDT o WFG. Su aplicación a otros problemas reales del campo de la ingeniería del software (planificación de proyectos, generación automática de casos de prueba, etc.) es la primera de estas líneas.

Por otra parte, se han diseñado extensiones grid de varias de las metaheurísticas (GrEA y assNSGA-II), pero existen otras técnicas, como búsqueda dispersa o colonias de hormigas, para las no existen modelos eficientes que aprovechen la enorme capacidad de cómputo que proporcionan este tipo de sistemas. Esto supone una línea clara de trabajo que se abre en esta tesis.

Parte V

Apéndices

Apéndice A

Relación de publicaciones que sustentan la tesis doctoral

En este apéndice se presenta el conjunto de trabajos que han sido publicados como fruto de las investigaciones desarrolladas a lo largo de esta tesis doctoral. Estas publicaciones avalan el interés, la validez, y las aportaciones de esta tesis doctoral en la literatura, ya que estos trabajos han aparecido publicados en foros de prestigio y, por tanto, han sido sometidos a procesos de revisión por reconocidos investigadores especializados. En la Figura A.1 se muestra un esquema de estas publicaciones. A continuación se muestran las referencias de todas las publicaciones.

Revistas indexadas en el ISI JCR:

[1] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation (en prensa)*, 2008.

[2] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems (en prensa)*, 2008.

[3] 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 (en segunda revisión con cambios menores)*, 2008.

[4] A. J. Nebro, G. Luque, F. Luna, and E. Alba. DNA fragment assembly using a grid based genetic algorithm. *Computers & Operations Research*, 35(9):2776 – 2790, 2008.

[5] A. J. Nebro, E. Alba, and F. Luna. Multi-objective optimization using grid computing. *Soft Computing*, 11(6):531 – 540, 2007.

[6] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, P. Bouvry, and L. Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. *Computer Communications*, 30(4):685 – 697, 2007.

[7] F. Luna, A. J. Nebro, and E. Alba. Observations in using grid-enabled technologies for solving multi-objective optimization problems. *Parallel Computing*, 32(5-6):377 – 393, June 2006.

[8] E. Alba, F. Luna, and A. J. Nebro. Parallel heterogeneous genetic algorithms for continuous optimization. *Parallel Computing*, 30(5-6):669 – 719, May/June 2004.

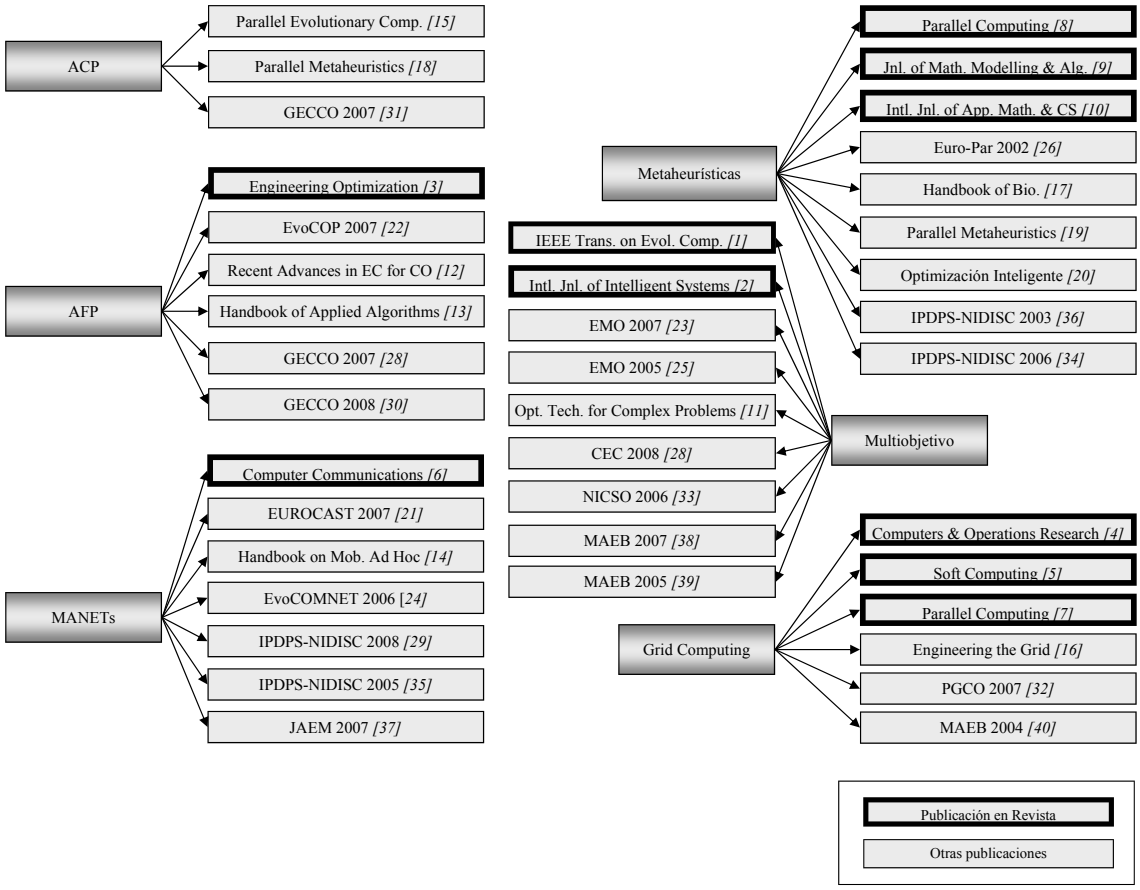


Figura A.1: Esquema de las publicaciones que avalan el trabajo realizado en esta tesis doctoral.

Revistas internacionales:

[9] E. Alba, G. Luque, and F. Luna. Parallel Metaheuristics for Workforce Planning. *Journal of Mathematical Modelling and Algorithms*, 6(3): 509 – 528, 2007.

[10] 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): 101 – 117, 2004.

Capítulos de libro:

[11] A. J. Nebro, J. J. Durillo, F. Luna, and E. Alba. Evaluating New Advanced Multiobjective Metaheuristics. In *Optimization Techniques for Solving Complex Problems*, Wiley (en prensa), 2008.

[12] F. Luna, E. Alba, A. J. Nebro, and S. Pedraza. Search intensification in metaheuristics for solving the automatic frequency problem. In *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, Studies in Computational Intelligence, Springer (en prensa), 2008.

[13] F. Luna, E. Alba, A. J. Nebro, P. Mauroy, and S. Pedraza. Applying evolutionary algorithms to solve the automatic frequency planning problem. In A. Nayak and I. Stojmenovic, editors, *Handbook of Applied Algorithms: Solving Scientific, Engineering and Practical Problems*. Wiley (en prensa), 2008.

[14] F. Luna, B. Dorronsoro, A. J. Nebro, E. Alba, and P. Bouvry. Multiobjective metaheuristics to optimize the broadcasting in MANETs. In L.T. Yang and M. Denko, editors, *Handbook on Mobile Ad Hoc and Pervasive Communications*. American Scientific Publishers (en prensa), 2008.

[15] F. Luna, A. J. Nebro, and E. Alba. Parallel evolutionary multiobjective optimization. In N. Nedjah, E. Alba, and L. de Macedo, editors, *Parallel Evolutionary Computations*, volume 22 of *Studies in Computational Intelligence*, chapter 2, pages 33 – 56. Springer, 2006.

[16] A. J. Nebro, E. Alba, and F. Luna. Observations in using grid technologies for multi-objective optimization. In B. Di Martino, J. Dongarra, A. Hoisie, L. T. Yang, and H. Zima, editors, *Engineering The Grid: Status and Perspective*, chapter 2, pages 27 – 39. American Scientific Publishers, 2006.

[17] E. Alba, F. Chicano, F. Luna, G. Luque, and A. J. Nebro. Advanced evolutionary algorithms for training neural networks. In S. Olariu and A. Y. Zomaya, editors, *Handbook of Bioinspired Algorithms and Applications*, pages 453 – 467. CRC Press, 2006.

[18] A. J. Nebro, F. Luna, E-G. Talbi, and E. Alba. Parallel multiobjective optimization. In E. Alba, editor, *Parallel Metaheuristics*, pages 371 – 394. Wiley, 2005.

[19] F. Luna, E. Alba, and A. J. Nebro. Parallel heterogeneous metaheuristics. In E. Alba, editor, *Parallel Metaheuristics*, pages 395 – 422. Wiley, 2005.

[20] E. Alba, J. F. Chicano, C. Cotta, B. Dorronsoro, F. Luna, G. Luque, and A. J. Nebro. Metaheurísticas secuenciales y paralelas para optimización de problemas complejos. In *Optimización Inteligente. Ténicas de Inteligencia Computacional para Optimización*, pages 185 – 214. Servicio de Publicaciones de la Universidad de Málaga, 2004.

Congresos internacionales publicados en la serie *Lecture Notes in Computer Science*:

[21] E. Alba, A. Cervantes, J. A. Gómez, P. Isasi, M. D. Jaraíz, C. León, C. Luque, F. Luna, G. Miranda, A.J. Nebro, R. Pérez, and C. Segura. Metaheuristics approaches for optimal broadcasting design in metropolitan MANETs. In *Eleventh International Conference on Computer Aided Systems Theory (EUROCAST 2007)*,volume 4739 of *LNCS* pages 262 – 263, 2007.

[22] F. Luna, E. Alba, A. J. Nebro, and S. Pedraza. Evolutionary algorithms for real-world instances of the automatic frequency planning problem in GSM networks. In *Seventh European Conference on Evolutionary Computation in Combinatorial Optimization (EVOCOP 2007)*, volume 4446 of *LNCS*, pages 108 – 120, 2007.

[23] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *LNCS*, pages 126 – 140. Springer, 2007.

[24] F. Luna, A. J. Nebro, B. Dorronsoro, E. Alba, P. Bouvry, and L. Hogie. Optimal broadcasting in metropolitan MANETs using multiobjective scatter search. In *Applications of Evolutionary Computing: EvoCOMNET 2006*, volume 3907 of *LNCS*, pages 255 – 266, 2006.

[25] A. J. Nebro, F. Luna, and E. Alba. New ideas in applying scatter search to multiobjective optimization. In C.A. Coello, A. Hernández, and E. Zitler, editors, *Third International Conference on Evolutionary MultiCriterion Optimization, EMO 2005*, volume 3410 of *LNCS*, pages 443 – 458. Springer, 2005.

[26] A. J. Nebro, E. Alba, F. Luna, and J. M. Troya. .NET as a platform for implementing concurrent objects. In *Euro-Par 2002 Parallel Processing*, volume 2400 of *LNCS*, pages 125–130, 2002.

Congresos internacionales:

[27] F. Luna, C. Estébanez, C. León, J. M. Chávez 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. En *Genetic and Evolutionary Computation Conference (GECCO 2008)*, 2008 (en prensa).

[28] J. J. Durillo, A. J. Nebro, C. Coello, F. Luna, and E. Alba. A Comparative Study of the Effect of Parameter Scalability in Multi-Objective Evolutionary Algorithms. En *IEEE Congress on Evolutionary Computation (CEC 2008)*, 2008 (en prensa).

[29] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba. A study of master-slave approaches to parallelize NSGA-II. En *IPDPS-NIDISC'08*, 2008 (en prensa).

[30] F. Luna, C. Blum, E. Alba, and A. J. Nebro. ACO vs EAs for solving a real-world frequency assignment problem in GSM networks. En *Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 94 – 101, 2007.

[31] A. J. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, and J. J. Durillo. Optimal antenna placement using a new multi-objective CHC algorithm. En *Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 876 – 883, 2007.

[32] B. Dorronsoro, D. Arias, F. Luna, A. J. Nebro, and E. Alba. A grid-based hybrid cellular genetic algorithm for very large instances of the VRP. En *Parallel and Grid Computing for Optimization, PGCO 2007 - HPCS 07*, pages 759 – 765, 2007.

[33] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. En D. A. Pelta and N. Krasnogor, editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25 – 36, 2006.

[34] E. Alba, G. Luque, and F. Luna. Workforce planning with parallel algorithms. En *IPDPS-NIDISC'06*, pages 1 – 8, 2006.

[35] E. Alba, P. Bouvry, B. Dorronsoro, F. Luna, and A. J. Nebro. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. En *Nature Inspired Distributed Computing (NIDISC) workshop of the International Parallel and Distributed Processing Symposium (IPDPS)*, page 192b, 2005.

[36] E. Alba, F. Luna, and A. J. Nebro. Parallel heterogeneous genetic algorithms for continuous optimization. En *IPDPS-NIDISC'03*, page 147, 2003.

Congresos nacionales:

[37] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, C. León, G. Miranda, and C. Segura. Metaheurísticas multiobjetivo para optimizar el proceso de difusión en MANETs metropolitanas. En *I Jornadas de Algoritmos Evolutivos y Metaheurísticas (JAEM'07)*, pages 229 – 236, 2007.

[38] A. J. Nebro, F. Luna, B. Dorronsoro, and J. J. Durillo. Un algoritmo multiobjetivo basado en búsqueda dispersa. En *Quinto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2007)*, pages 175 – 182, 2007.

[39] A. J. Nebro, F. Luna, E. Alba, and A. Beham. Un estudio de la aplicación del algoritmo de búsqueda dispersa a optimización multi-objetivo. En *Cuarto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2005)*, 2005.

[40] A. J. Nebro, E. Alba, and F. Luna. Optimización multi-objetivo y computación grid. En *Actas del Tercer Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB'04)*, pages 365 – 372, 2004.

Apéndice B

English Summary

The design of more efficient algorithms in order to solve complex problems has been one of the most important aspects in computer science research. The aim is the development of new methods that are able to solve complex problems with low computational efforts, outperforming the current algorithms. In this context, the research activity in exact, ad-hoc heuristics, and metaheuristic algorithms is increasing nowadays.

The main advantage of using exact algorithms is that they always find the global optimum for any problem. However, they have an important drawback: in real problems (NP-hard in most cases) their execution time grows in an exponential way with the size of the problem. On the other hand, the ad-hoc heuristic algorithms usually are quite fast, but the quality of the solutions found is far from the optimum. Another drawback of the ad-hoc heuristics is that they are difficult to design in some problems. The metaheuristic algorithms offer an appropriate balance between both extremes: they are generic methods that offer high quality solutions (global optimum in many cases) in a reasonable time.

Telecommunication networks are an important source of optimization problems. Engineers have to tackle many optimization problems that can not be solved with exact techniques because a solution is required in short time. Thus, it is possible to apply metaheuristic algorithms to these problems in order to provide the experts with a good balance between quality and efficiency.

In this PhD dissertation we have applied metaheuristic techniques to a set of real-world optimization problems found in the telecommunications domain. We have used advanced techniques for addressing the two main features of this kind of problems: on the one hand, many of them are multiobjective in nature (that is, several functions have to be optimized at the same time) and, on the other hand, they involve tasks demanding high computational resources. In this context, the proposed methods are engineered to deal with multiobjective optimization problems and they have been enabled to run on grid computing systems in order to profit from the large computing capacity that these systems are able to provide.

B.1. Organization

This thesis is organized in five parts. In the first one, the principles of optimization problems arising in the telecommunication domain and metaheuristics are presented. In the second part we detail the formulation of the three problems tackled in this thesis. The methodology proposals and the results obtained after the application of metaheuristic algorithms to the problems are presented

in the third part. The fourth part is devoted to showing the main conclusions and future work of the thesis. Finally, in the fifth part two appendices can be found. Next, we detail the content of the chapters.

- **Chapter 1: Introduction.** This chapter presents an outline of the dissertation including objectives, phases, contributions, and organization.

- **Chapter 2: Optimization and Telecommunication Networks.** We analyze the optimization problems that arise in telecommunication networks and put the work of this PhD dissertation in context. Then, we provide the reader with an overview of the three problems tackled: the automatic cell planning problem (ACP), the automatic frequency planning problem (AFP), and the optimal broadcasting problem in mobile ad hoc networks.

- **Chapter 3: Metaheuristics.** This chapter introduces the metaheuristic techniques, classifies them, and gives a formal definition. We also give a background on multiobjective optimization concepts since two of the tackled problems (the ACP and the optimal broadcasting problems) fall into this type of optimization tasks. Since this work proposes several algorithmic extensions to enable metaheuristics to run in grid computing systems, a description of grid technologies is provided as well.

- **Chapter 4: The Automatic Cell Planning Problem.** A mathematical formulation of the automatic cell planning problem is given in this chapter.

- **Chapter 5: The Automatic Frequency Planning Problem**. This chapter proposes a new mathematical formulation for the AFP problem. It uses a very accurate interference information directly imported from real world GSM frequency planning as currently conducted in the industry.

- **Chapter 6: Optimal Broadcasting Strategy in MANETs.** The problem of optimally tuning a broadcasting protocol for metropolitan MANETs is formulated in this chapter.

- **Chapter 7: Methodology Proposals**. This chapter describes the proposals for applying the selected metaheuristic algorithms to each problem. The contributions of this chapter are manifold: the new state-of-the-art multiobjective algorithms ssNSGA-II, AbYSS, and MOCell, and the extension for enabling them to run on grid computing systems.

- **Chapter 8: Solving the ACP Problem** This chapter presents and analyzes the results obtained after using the algorithmic proposals to solve the ACP problem.

- **Chapter 9: Solving the AFP Problem** The results of solving the automatic frequency planning problem with the proposed metaheuristics are evaluated in this chapter.

- **Chapter 10: Solving the Optimal Broadcasting Problem in MANETs** In this chapter, we analyze the results of the proposed algorithms when solving the problem of optimal broadcasting in MANETs.

- **Appendix A: Publications.** This appendix presents the publications of the PhD student that are related to the PhD dissertation and support its quality.

- **Appendix B: English Summary.** A summary of this thesis in English.

B.2. Optimization Problems in Telecommunication Networks

In the second chapter of this thesis we present a general view of the optimization problems that arise in telecommunication networks. These problems are numerous and their satisfactory resolution has played an important role in the development and generalized utilization of this kind of systems [219]. Many of these optimization problems are related to the design and the operation of the underlying communication network that is used to transfer the information. We have selected three such problems to solve in this thesis.

Cellular phone networks are, with no doubt, the telecommunication systems which have received more attention. The most important contribution of optimization here is to improve the use of the scarce resources (e.g., radio spectrum, antennae, etc.) and to increase the quality of the services provided (e.g., bandwidth, transmission delay, etc.). Two optimization problems have been considered here, both related to the radio network planning of the GSM cellular system [178]: ACP and AFP.

The third optimization problem addressed in this PhD dissertation comes from the mobile ad hoc networking field, a novel communication system in which a set of communicating devices which are able to spontaneously interconnect without any pre-existing infrastructure. In these networks, broadcasting becomes an operation of capital importance for the own existence and operation of the network. We have formulated the fine-tuning of broadcasting strategies as an optimization problem so as to reach the best configuration possible for a given target scenario.

B.3. Tackled Problems

In the second part we present in detail the three telecommunication network problems that we have selected to solve with metaheuristic techniques. These problems are the automatic cell planning problem, the automatic frequency planning problem, and the optimal broadcasting design in MANETs. In the following sections we present a brief description of the three problems.

B.3.1. Automatic Cell Planning

The automatic cell planning problem (or Coverage Problem) [217] can be described as follows. Given an area where the service has to be guaranteed, determine where to install the Base Transceiver Stations (BTSs) and select their configurations so that each point (or each user) in the service area receives a sufficiently high signal. Since the cost associated to each BTS may depend on its location and configuration, a typical goal is that of minimizing the total antenna installation cost while guaranteeing service coverage.

As discussed, cell planning involves two optimization tasks. First, a number of sites for the BTSs have to be chosen from a set of candidate sites. Once these sites are selected, the BTSs have to be configured so as to reach a minimum coverage and capacity. Efficiently configuring the BTSs (power, tilt, azimuth, etc.) is a difficult task not only because of the large number of potential settings but also due to fact that changing the configuration of a BTS may affect the area served by other BTSs. It has been defined as a multiobjective problem with three objectives (minimizing the number of sites, maximizing the traffic hold, and minimizing the interferences) and two constraints (a minimum area must be covered, and the handover procedure has to be guaranteed).

B.3.2. Automatic Frequency Planning

Frequency planning is the last step in the layout of a GSM network. Once the sites for the BTSs are selected and the sector layout is decided, the number of transceivers (TRXs) to be installed per sector has to be fixed. This number depends on the traffic demand that the corresponding sector has to support. Frequency planning lies in the assignment of a channel (a frequency) to every TRX [79]. The optimization problem arises because the usable radio spectrum is generally very scarce and, consequently, frequencies have to be reused by many TRXs in the network.

However, the multiple use of a same frequency may cause interferences that may reduce the quality of service (QoS) down to unsatisfactory levels. Indeed, significant interference may occur if the same or adjacent channels are used in neighboring, overlapping cells. The point here is that computing this level of interference is a difficult task which depends not only on the channels, but also on the radio signals and the properties of the environment. The more accurate the measure of the interference in a given GSM network, the higher the quality of the frequency plan that can be computed for this network. Several ways of quantifying this interference exist, ranging from theoretical methods to extensive measurements [142]. They all result in a so-called *interference matrix*, denoted by M. Each element $M(i, j)$ of M indicates the degradation of the network quality if cells i and j operate on the same frequency. This is called *co-channel interference*. Apart from co-channel interference there may exist a so-called *adjacent-channel interference*, which occurs when two TRXs operate on adjacent channels (i.e., one TRX operates on channel f and the other on channel $f + 1$ or $f - 1$). An accurate interference matrix is therefore an essential requirement for frequency planning because the ultimate goal of any frequency assignment algorithm will be to minimize the sum of the interferences.

B.3.3. Optimal Broadcasting Design in MANETs

Devices in MANETs are usually laptops, PDAs, or mobile phones, equipped with network cards featuring wireless technologies such as Bluetooth and/or IEEE802.11 (WiFi). This implies that a) devices communicate within a limited range and b) devices can move while communicating; thus, the network topology may change quickly and unpredictably. This dynamic behavior constitutes one of the main obstacles for performing efficient communications on such networks.

Broadcasting, i.e., the process in which a source node sends a message to all the nodes in the network, is a common operation at the application level. It is also widely used for solving many network layer problems constituting, for example, the basic mechanism for many routing protocols. In a given MANET, due to host mobility, broadcasting is expected to be performed very frequently. Hence, choosing an appropriate broadcasting strategy will result in a major impact in network performance.

Our approach here is to reach the best possible broadcasting strategy for a given particular scenario [11]: optimally tuning the broadcasting service for a set of networks and for a particular category of broadcast messages. This optimal tuning has been formulated as an optimization problem in which multiple objectives have to be satisfied at the same time, none of them being secondary (multiobjective optimization). The objectives are maximizing the number of devices reached (coverage), minimizing the network use (bandwidth), and minimizing the duration of the process.

B.4. Metaheuristics

A *metaheuristic* can be defined as a high level strategy that combines different methods for exploring the search space [29, 100]. It is a general template that must be filled with problem-specific knowledge (solution representation, operators, etc.) and can tackle problems with very large search spaces. We can classify the metaheuristics in two classes, depending on the number of solutions they manipulate at each step: *trajectory based* and *population based* metaheuristics. The techniques in the first class work with one solution that is modified in each step. On the other hand, the population based metaheuristics work with a set of solutions. Some examples of trajectory based metaheuristics are: simulated annealing (SA), tabu search (TS), GRASP, variable neighborhood search (VNS), and iterated local search (ILS). Some examples of population based metaheuristics are: evolutionary algorithms (EA), estimation of distribution algorithms (EDA), scatter search (SS), ant colony optimization (ACO), and particle swarm optimization (PSO).

There are two major features of the previously defined problems that have to be taken into consideration when using metaheuristics. On the one hand, both the ACP and the optimal broadcasting problems are multiobjective in nature, i.e., they have several (contradictory) functions which must be optimized simultaneously. And, on the other hand, they all involve tasks demanding high computational resources due to both the accurate formulation and the realistic data of the instances. These two characteristics have encouraged us to propose advanced strategies to address these challenging optimization problems. We have used two approaches: multiobjective optimization techniques based on Pareto optimality and grid computing in order to reduce the computational times to affordable values. The two are discussed in the next sections.

B.4.1. Dealing with Multiple Objectives

Multiobjective optimization seeks to optimize several functions at the same time [44, 65]. Contrary to single objective optimization, the solution of a MOP is not a single solution, but a set of solutions known as *Pareto optimal set*, which is called Pareto border or *Pareto front* when it is plotted in the objective space. Any solution of this set is optimal in the sense that no improvement can be made on a component of the objective vector without worsening at least another of its components.

The main goal in the resolution of a multiobjective problem is to obtain the set of solutions within the Pareto optimal set and, consequently, the Pareto front. However, when metaheuristics are applied, the goal becomes to obtain a "good" approximation to the Pareto front, i.e., a set of nondominated solutions having two properties: convergence to the true Pareto front and homogeneous diversity of solutions along this front. The first property ensures that we are dealing with optimal solutions, while obtaining a uniform-spaced set of solutions indicates that we have carried a good exploration of the search space, so we are not losing valuable information.

B.4.2. Enabling Metaheuristics to Run on Grids

Even when using metaheuristic algorithms, the time required to find a solution for a given optimization problem can be very long. In this context, parallelism can help not only in reducing this time, but also in finding better solutions (better sampling of the solution space). In most real-world optimization problems, such as those tackled here, the total computing time can rise up to hundreds of days, therefore they cannot be addressed in normal clusters of machines. Grid computing systems (or Grids) [26] have emerged as a platform that provides the equivalent computing power of hundreds and thousands of computers, thus enabling to execute in a reasonable amount of time algorithms which otherwise would be considered as unfeasible.

In order for metaheuristics to profit from the high computational capacity of a grid computing system, they have to manage the particular features of these systems: heterogeneity of the resources, dynamic availability, fault tolerance, or high communication latency. Classic communication libraries such as PVM/MPI or mechanisms based on remote procedure call (CORBA, Java RMI) are not appropriate to be directly used as programming tools for grid-based optimization algorithms. A specific middleware such as Globus [89, 88] or Condor [238] is required. Once the grid middleware provides the optimization algorithm designer with the proper tools for managing all the issues of the grid platform, this designer has to engineer a model that allows the metaheuristic to take advantage of the computational resources.

B.4.3. Advanced Metaheuristic Proposals

Two algorithms have been chosen for solving the three problems: a steady state genetic algorithm (ssGA) [20] and a scatter search approach (SS) [98, 97, 99]. We have selected a ssGA because it is well known that this kind of metaheuristics (Evolutionary Algorithms) performs well on complex search spaces, thus reaching very accurate solutions efficiently. As for SS, this is a rather novel metaheuristic which has proven to be very effective for many optimization problems, but it has never been faced to such a real world problem as those tackled here. So our goal is to investigate the capacity of SS for solving these problems. Since both the ACP and the optimal broadcasting problems have been formulated as multiobjective problems, we have engineered the multiobjective extensions of ssGA and SS: ssNSGA-II (steady state NSGA-II) and AbYSS (Archive-based hYbrid Scatter Search), respectively, in order to solve these two problems. Additionally, we have developed a specialized algorithm for each problem: parallel Evolution Strategy, called pPAES, for the ACP problem, an ACO (Ant Colony Optimization) algorithm for the AFP problem, and a cellular GA, named MOCell, for the optimal broadcasting problem in MANETs. In summary, each problem is addressed as follows:

Problem	Algorithm 1	Algorithm 2	Algorithm 3
ACP	ssNSGA-II	AbYSS	pPAES
AFP	ssGA	SS	ACO
MANETs	ssNSGA-II	AbYSS	MOCell

We have designed an extension of several of the previously proposed solvers in order to enable them to run on grid computing systems. Since evaluating any given tentative solution of any of the three problems is computationally expensive, the time required for any algorithm to run on a single machine can rise up to unaffordable values. In particular, we have developed the grid extensions of ssGA in the monoobjective case and ssNSGA-II in the multiobjective one. They are named, respectively, GrEA, and assNSGA-II.

B.5. Solving the ACP Problem with the Proposed Algorithms

We have used a very accurate formulation as well as three real-world instances for addressing the ACP problem. We have also included a review of the literature in order to look for other approaches applied to solve this kind of problems. In this sense, the three proposed metaheuristics, ssNSGA-II, AbYSS, and pPAES, are a contribution in the field since they have never been used in this particular problem.

B.5.1. Solution Encoding and Basic Search Operators

We have first introduced both the solution representation and the basic search operators for the metaheuristic algorithms. The tentative ACP solutions manipulated by the solvers encode all potential information of the network. We use a multilevel encoding: level 1 decides the activation of sites, level 2 decides the number and the type of antennas, and level 3 handles the parameters of the BTS. When a site is used, one or more antennas are activated. When an omnidirectional antenna is chosen, we settle the power and the tilt parameters. Moreover, when a directive antenna is used, the azimuth parameter is initialized.

The basic search operators for all the algorithms are based on the geographic crossover and multilevel mutation. The geographical crossover works at level 1 of the encoding hierarchy. It exchanges sites that are located within a given radius around a randomly chosen site. In this context, the crossover operator is non-destructive: the offspring inherit good properties of the parents. The site representing the center and the radius are chosen randomly. The information of every site for the levels 2 and 3 of the encoding is inherited from the parents.

The mutation operator acts at all levels of the gene hierarchy. When a mutation occurs, it may act at one level at a time. The mutation type is chosen among station activation toggling, transmitter power tuning, BS tilt tuning, BS azimuth tuning and BS diagram tuning. The mutation is chosen uniformly among the considered five different mutation operators.

B.5.2. Discussion of the Results

Three set of experiments have been conducted. In the first one, we have tackled the three instances, Arno 1.0, Arno 3.0 and Arno 3.1, with our proposed metaheuristics, ssNSGA-II, AbYSS and PAES (the algorithm pPAES is based on). For means of comparison, the two state-of-the-art multi-objective optimization algorithms are used: NSGA-II and SPEA2. The results show that ssNSGA-II is the best performing approach in the three given instances, even outperforming NSGA-II and SPEA2. The results of AbYSS can be considered to be in a preliminary stage because scatter search methods have never been applied to solve this kind of problems. However, differences with the rest of the algorithms is kept regardless of the instance complexity, which is a promising starting point for future research. The PAES algorithm is clearly the worst algorithm, an expected result since it does not use a recombination operator and therefore has its search capabilities diminished.

The next set of experiments is devoted to pPAES. We have studied both the parallel behavior and the search capabilities of the algorithm for different parameter settings. In the first case, the results show that pPAES is able of getting major reductions in the execution time when a few number of processors is involved in the computation. The speedup decreases, however, when the number of processors increases. The synchronous unidirectional ring topology causes this behavior. However, the newly designed search model is more accurate: the quality of the solutions reached by pPAES outperform that of PAES, i.e., pPAES better explores the search space of the ACP problem.

The last set of experiments analyzes the results of solving the ACP problem using grid computing system. The proposed algorithm, assNSGA-II, extends ssNSGA-II to be deployed on this kind of system. As for the parallel performance of the algorithm, it is able to reduce the computation time from two days down to one hour by using a grid platform composed of more than 300 processors. From the point of view of solution quality, the new proposal is also more effective because, by evaluating the same number of solutions (same sampling of the search space), assNSGA-II computes more accurate solutions than ssNSGA-II.

B.6. Solving the AFP Problem

Solving the AFP problem, i.e., assigning the limited frequency spectrum available to the TRXs of a cellular network, is a crucial problem for current operators. We have tackled three real-world instances that correspond to networks deployed in three cities in the USA: Seattle, Denver, and Los Angeles. Three metaheuristic algorithms have been proposed to address this optimization problem: a steady state genetic algorithm (ssGA), a scatter search method (SS), and an ant colony optimization algorithm (ACO).

B.6.1. Solution Encoding and Basic Search Operators

A solution to the AFP problem is obtained by assigning one valid frequency to each TRX. A solution is therefore encoded as an array of integer values p, where p_i is the frequency assigned to transceiver i. That is, the solutions manipulated are tentative frequency plans of the given AFP problem instance.

As for the search operators, we have used *uniform crossover* (UX) in which every allele of the offspring (i.e., the frequency of each TRX) is chosen randomly from one of the two parents with a probability of 0.5. The mutation operator applied is the *random mutation* in which the frequencies of a set of randomly chosen TRXs of the solution are reassigned with a random valid frequency. Note that the two operators always assign valid frequencies to each TRX and no repair step is required.

B.6.2. Results

The three proposed algorithms have successfully solved the AFP problem considered in this PhD dissertation. The results show that the ACO approach is the most accurate solver in short term executions, reaching high quality solution quickly. ACO is the best performing algorithm in the smaller instance used (Seattle). However, it is outperformed by ssGA when longer executions are considered in the Denver instance (larger size). In this scenario, ssGA has always reached the best frequency plans. Finally, SS is the algorithm that produces the worst results among the three. Since this is the first time a scatter search method is applied to this kind of optimization problem, the results are considered to be promising due to two observed facts: first, the search does not get stuck when SS runs for long times and, second, the algorithm scales well with different problem sizes.

We have also addressed the AFP problem using a grid-enabled metaheuristics, which is called GrEA. The contribution of this algorithm to this problem is twofold. On the one hand, it is able to reduce the execution time from more than five days to one hour an a half, meaning that the algorithm can profit from the high-end computational resources provided by our grid system composed of 300 processors. On the other hand, GrEA is also more effective than its sequential counterpart, ssGA, i.e., by using the same numerical effort, the former reaches more accurate frequency plans than the latter. This allows us to conclude that the newly proposed search model used to deploy the algorithm on the grid is better.

B.7. Solving the Optimal Broadcasting Problem in MANETs

Optimally tuning a broadcasting protocol in MANETs is a multiobjective problem in which three functions have to be optimized at the same time: maximizing the coverage of the broadcasting procedure, minimizing the bandwidth used, and minimizing the duration of the process. In order to address this problem we have used three algorithms: ssNSGA-II, AbYSS, and cMOGA/MOCell.

We have shown the effectiveness of these approaches by comparing them with the two algorithms of the state-of-the-art in multiobjective optimization: NSGA-II and SPEA2.

B.7.1. Solution Encoding and Basic Search Operators

A solution for this optimization problem is composed of the five parameters of the given broadcasting protocol that is to be tuned. Here, the broadcasting strategy is DFCN and the parameters are: *MinGain*, *[lowerBoundRAD, upperBoundRAD]*, *proD* y *safeDensity*. They are encoded as a floating point vector and define the search space of the problem.

The search operators in all the algorithms are based on *Simulated Binary Crossover* (SBX) and *Polynomial Mutation* [66]. SBX simulates the working principle of the single-point crossover on binary genotypes, while the Polynomial mutation is characterized by a polynomial probability distribution used to perturb the real valued genes.

B.7.2. Discussion of the Results

The first study that we have developed lies in characterizing the optimization problem itself: number of nondominated solutions and shape of the Pareto front, execution times, etc. The algorithm used here is cMOGA, the first metaheuristic used to tackle this optimization problem. The results show that the execution times are longer than one day. The search space of the problems is also highly complex, since cMOGA finds difficulties to fill a Pareto front of 100 nondominated solutions. Therefore, it can be considered as a challenging problem for the solvers.

The next step is that of systematically solving the problem with all the proposals. The results show that our solvers ssNSGA-II and MOCEll, together with SPEA2, have reached the most accurate Pareto fronts for the problem. They three outperform NSGA-II, the most well-known algorithm in the multiobjective research community.

Finally, we have addressed the optimal broadcasting problem in MANETs by using grid computing technologies. The algorithm utilized is assNSGA-II. Two issues have driven this study: solution quality and parallel behavior. In the first case, assNSGA-II has the same search capabilities as the best performing sequential algorithms used for solve the problem. The contribution of this proposal is based on its ability of taking advantage of the huge computational power offered by a grid computing platform. Indeed, the algorithm has reduced the wall-clock time from one day to just seven minutes in the best execution (20 minutes on average).

B.8. Conclusions and Future Work

This PhD dissertation is devoted to solving real world problems in telecommunication networks by using metaheuristics. Two main features of these problems exist that have guided this work. First, they are engineering problems in which several functions have to be optimized at the same time, i.e., they are multiobjective problems. Second, we are tackling not only real world problems but also real world instances, what involves tasks demanding high computational resources, even when heuristics algorithms are used. We have therefore proposed two kind of advances techniques in order to tackle these problems: Pareto-based techniques for dealing with multiobjective problems, and grid computing systems for reducing the computing times to affordable values.

The three problems addressed in this thesis are the automatic cell planning (ACP) and the automatic frequency planning problem (AFP), both coming from the cellular networks industry, and the optimal broadcasting design in mobile ad hoc networks (MANETs), belonging to the field of wireless networking. For each of these three problems, an accurate mathematical definition has been proposed which is based on concepts and models used in the industry. Additionally, we have

realistic data for the three problems, three real world instances concretely, for which their solution is of great interest and it poses a great challenge to any optimization algorithm.

Next, we have selected two techniques which the three problems have been addressed with: steady state evolutionary algorithms, ssGA, and scatter search, SS. Since both the ACP and the optimal broadcasting problems have been formulated as multiobjective problems, we have engineered the multiobjective extensions of ssGA and SS, ssNSGA-II (steady state NSGA-II) and AbYSS (Archive-based hYbrid Scatter Search), in order to solve these two problems. Additionally, we have developed an especialized algorithm for each problem: parallel Evolution Strategy, called pPAES, for the ACP problem, an ACO (Ant Colony Algorithm) algorithm for the AFP problem, and a cellular GA, named MOCell, for the optimal broadcasting problem in MANETs.

The results revealed that steady state evolutionary algorithms are the most suitable approach for addressing real world problems involving tasks which demand high computational resources, not only in monoobjective optimization, but also in the multiobjective case (ssNSGA-II have always outperformed NSGA-II). Moreover, their evolutionary cycle allows for an efficient extension which enables them to run on grid computing systems. GrEA and assNSGA-II are two clear examples. The truly interesting fact here is that both GrEA and assNSGA-II are not only able of reducing the execution times of ssGA and ssNSGA-II (their sequential counterparts) hundreds of times, but also the underlying search model which has emerged is more accurate (in some cases), i.e., it reaches improved solutions by using the same computational effort. As to the scatter search approach, this is the first time this algorithm is applied to this kind of problems and, therefore, the reported results are slightly worse. However, it has a very promising behavior because it is scalable, i.e., its accuracy does not decrease with larger instances, and it does not get stuck during long term executions. There is still room for scatter search to provide the tackled optimization problems with satisfactory solutions.

As future work, there are two main open research lines: real-world problem solving and metaheuristics. In the first case, the lessons learned from the three tackled problems can be applied to address other real-world problems, not only from the telecommunications domains, but also from other fields such as the search-based software engineering (e.g., software project scheduling). Finally, the design of new advanced metaheuristics to effectively and efficiently solving multiobjective optimization problems, as well as engineering new models to deploy metaheuristics procedures on grid computing systems, such as scatter search, ant colony optimization or differential evolution, are also open research lines.

Bibliografía

[1] 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.

[2] J. Abril, F. Comellas, F. Cortés, J. Ozón, and M. Vaquer. A multiagent system for frequency assignment in cellular radio networks. *IEEE Transactions on Vehicular Technology*, 49(5):1558 – 1565, 2000.

[3] A. Acan and A. Günay. An external memory supported ACO for the frequency assignment problem. In *Adaptive and Natural Computing Algorithms*, pages 365 – 368, 2005.

[4] M. Alabau, L. Idoumghar, and R. Schott. New hybrid genetic algorithms for the frequency assignment problem. In *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, pages 136 – 142, 2001.

[5] M. Alabau, L. Idoumghar, and R. Schott. New hybrid genetic algorithms for the frequency assignment problem. *IEEE Transactions on Broadcasting*, 48(1):27 – 34, 2002.

[6] E. Alba. *Análisis y Diseño de Algoritmos Genéticos Paralelos Distribuidos*. PhD thesis, University of Málaga, 1999.

[7] E. Alba. Evolutionary algorithms for optimal placement of antennae in radio network design. In *IPDPS-NIDISC'04*, page 168, 2004.

[8] E. Alba, editor. *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, 2005.

[9] E. Alba, P. Bouvry, B. Dorronsoro, F. Luna, and A. J. Nebro. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. In *Nature Inspired Distributed Computing (NIDISC) workshop of the International Parallel and Distributed Processing Symposium (IPDPS)*, page 192b, 2005.

[10] E. Alba and F. Chicano. On the behavior of parallel genetic algorithms for optimal placement of antennae in telecommunications. *International Journal of Foundations of Computer Science*, 16(2):343 – 359, 2005.

[11] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, P. Bouvry, and L. Hogie. A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs. *Computer Communications*, 30(4):685 – 697, 2007.

[12] 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):101 – 117, 2004.

[13] E. Alba, G. Molina, and F. Chicano. Optimal placement of antennae using metaheuristics. In *Numerical Methods and Applications, 6th International Conference (NMA 2006), Revised Papers*, LNCS 4310, pages 214 – 222, 2007.

[14] E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443 – 462, 2002.

[15] E. Alba and J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4):451 – 465, 2001.

[16] Z. Altman, J. M. Picard, S. Ben Jamaa, B. Fourestié, A. Caminada, T. Dony, J. F. Morlier, and S. Mourniac. New challenges in automatic cell planning of umts networks. In *2002 IEEE 56th Vehicular Technology Conference Proceedings*, pages 951 – 954, 2002.

[17] Z. Altman, J. M. Picard, S. Ben Jamaa, B. Fourestié, A. Caminada, T. Dony, J. F. Morlier, and S. Mourniac. Oasys: Ftr&d umts automatic cell planning tool. In *IEEE Antennas and Propagation Society International Symposium*, pages 338 – 341, 2002.

[18] E. Amaldi, A. Capone, and F. Malucelli. Optimizing base station siting in UMTS networks. In *Proceedings 53rd IEEE Conference on Vehicular Technology*, pages 2828 – 2832, 2001.

[19] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, 1996.

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

[21] L. Baduel, F. Baude, D. Caromel, A. Contes, F. Huet, M. Morel, and R. Quilici. *Grid Computing: Software Environments and Tools*, chapter Programming, Deploying, Composing, for the Grid. Springer, 2006.

[22] M. Baker, R. Buyya, and D. Laforenza. Grids and grid technologies for wide area distributed computing. *Software: Practice and Experience*, 32:1437 – 1466, 2002.

[23] 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.

[24] J. E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069 – 1072, 1990.

[25] R. P. Beausoleil. MOSS: Multiobjective scatter search applied to nonlinear multiple criteria optimization. *European Journal of Operational Research*, 169(2):426 – 449, March 2006.

[26] F. Berman, G.C. Fox, and A.J.G. Hey. *Grid Computing. Making the Global Infrastructure a Reality.* Communications Networking and Distributed Systems. Wiley, 2003.

[27] P. Björklund, P. Värbrand, and D. Yuan. Optimized planning of frequency hopping in cellular networks. *Computers and Operations Research*, 32(1):169 – 186, 2005.

[28] C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 34(2):1161 – 1172, 2004.

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

[30] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, 76:73 – 93, 1998.

[31] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22:251 – 256, 1979.

[32] H. J. Bremermann. *Self-Organizing Systems*, chapter Optimization Through Evolution and Recombination, pages 93–106. Spartan Books, Washington DC, 1962.

[33] L. Brunetta, B. Di Chiara, F. Mori, M.Ñonato, R. Sorrentino, M. Strappini, and L. Tarricone. Optimization approaches for wireless network planning. In *URSI 2004 International Symposium on Electromagnetic Theory*, pages 182 – 184, 2004.

[34] S. Cahon, E-G. Talbi, and N. Melab. A parallel and hybrid multi-objective evolutionary algorithm applied to the design of cellular networks. In *MELECON 2006. 2006 IEEE Mediterranean Electrotechnical Conference*, pages 803 – 806, 2006.

[35] P. Calégari, F. Guidec, and P. Kuonen. A parallel genetic approach to transceiver placement optimisation. In *Proceedings of the SIPAR Workshop'96: Parallel and Distributed Systems*, pages 21 – 24, 1996.

[36] P. Calégari, F. Guidec, P. Kuonen, and D. Kobler. Parallel island-based genetic algorithm for radio network design. *Journal of Parallel and Distributed Computing*, 47(1):86 – 90, 1997.

[37] P. Calégari, F. Guidec, P. Kuonen, and F.Ñielsen. Combinatorial optimization algorithms for radio network planning. *Theoretical Computer Science*, 263:235 – 245, 2001.

[38] G. Cerri, R. De Leo, D. Micheli, and P. Russo. Base-station network planning including environmental impact control. In *Electrical Engineering and Electromagnetics VI. Sixth International Conference on Computational Methods for the Solution of Electrical and Electromagnetic Engineering Problems incorporating Electromagnetic Effects on Human Beings and Equipment Seminar. ELECTROCOMP VI*, pages 63 – 69, 2003.

[39] G. Cerri, R. De Leo, D. Micheli, and P. Russo. Base-station network planning including environmental impact control. *IEE Proceedings Communications*, 151(3):197 – 203, 2004.

[40] G. Cerri and P. Russo. Application of an automatic tool for the planning of a cellular network in a real town. *IEEE Transactions on Antennas and Propagation*, 54(10):2890 – 2901, 2006.

[41] B. Chamaret and C. Condevaux-Lanloy. Graph based modeling for automatic transmitter location in cellular network. In *Proceedings of the High Performance Computing HPC98 Special session Telepar'98*, pages 248 – 252, 1998.

[42] B. Di Chiara, M.Ñonato, M. Strappini, L. Tarricone, and M. Zappatore. Hybrid metaheuristic methods in parallel environments for 3G network planning. In *EMC Europe Workshop 2005, Electromagnetic Compatibility of wireless Systems*, pages 199 – 202, 2005.

[43] J. F. Chicano. *Metaheurísticas e Ingeniería del Software*. PhD thesis, University of Málaga, 2007.

[44] C. A. Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation Series. Springer, second edition, 2007.

[45] J. L. Cohon and D. H. Marks. A review and evaluation of multiobjective programming techniques. *Water Resources Research*, 11(2):208 – 220, 1975.

[46] G. Colombo. A genetic algorithm for frequency assignment with problem decomposition. *International Journal of Mobile Network Design and Innovation*, 1(2):102 – 112, 2006.

[47] F. Comellas and J. Ozón. Agentes distribuidos para la asignación de frecuencias hopping en redes celulares. In *Primer Congreso Español de Algoritmos Evolutivos y Bioinspirados*, pages 139 – 145, 2002.

[48] M. Conti, S. Giordano, G. Maselli, and G. Turi. MobileMAN: Mobile Metropolitan Ad Hoc Networks. In *Proc. of the Eight Int. IFIP-TC6 Conf.*, pages 194 – 205, 2003.

[49] A. Corberán, E. Fernández, M. Laguna, and R. Martí. Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53(4):427 – 435, 2002.

[50] COST231. Urban transmission loss models for mobile radio in the 900 and 1800 mhz bands. Technical report, European Cooperation in the Field of Scientific and Technical Research, 1991.

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

[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, 2003. Kluwer Academic Publishers.

[53] N.L. Cramer. A representation for the adaptive generation of simple sequential programs. In J. J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 183 – 187, 1985.

[54] J.-C. Créput, A. Koukam, T. Lissajoux, and A. Caminada. Automatic mesh generation for mobile network dimensioning using evolutionary approach. *IEEE Transactions on Evolutionary Computation*, 9(1):18 – 30, 2005.

[55] C. Crisan and H. Mühlenbein. The breeder genetic algorithm for frequency assignment. In *Parallel Problem Solving from Nature (PPSN V)*, pages 897 – 906, 1998.

[56] C. Crisan and H. Mühlenbein. The frequency assignment problem: a look at the performance of evolutionary search. In *Artificial Evolution. Third European Conference AE'97. Selected-Papers*, pages 263 – 273, 1998.

[57] W. Crompton, S. Hurley, and N. M. Stephens. Frequency assignment using a parallel genetic algorithm. In *Proc. 2nd IEE/IEEE Workshop on Natural Algorithms in Signal Processing (NASP'93)*, pages 26/1 – 26/8, 1993.

[58] W. Crompton, S. Hurley, and N. M. Stephens. A parallel genetic algorithm for frequency assignment problems. In *Proceedings of the IMACS/IEEE Conference on Signal Processing, Robotics and Neural Networks*, pages 81 – 84, 1994.

[59] 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*, pages 263–308, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[60] Carlos Gomes da Silva, Joao Clímaco, and José Figueira. A scatter search method for the bi-criteria multi-dimensional {0,1}-knapsack problem using surrogate relaxation. *Journal of Mathematical Modelling and Algorithms*, 3(3):183 – 208, January 2004.

[61] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, Londres, 1859.

[62] A. De Pasquale, N. P. Magnani, and P. Zanini. Optimizing frequency planning in the GSM system. In *IEEE 1998 Int. Conf. on Universal Personal Communications*, pages 293 – 297, 1998.

[63] K. Deb. *Optimization for Engineering Design*. Prentice-Hall, New Delhi, 1995.

[64] K. Deb. An efficient constraint handling mechanism method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311 – 338, 2000.

[65] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[66] K. Deb and R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[67] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849 – 858, 2000.

[68] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182 – 197, 2002.

[69] J. Demšar. Statistical comparison of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1 – 30, 2006.

[70] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartamento di Elettronica, Politecnico di Milano, 1992.

[71] M. Dorigo and T. Stützle. *Handbook of Metaheuristics*, volume 57 of *International Series In Operations Research and Management Science*, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, pages 251 – 285. Kluwer Academic Publisher, 2003.

[72] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

[73] R. Dorne and J.-K. Hao. An evolutionary approach for frequency assignment in cellular radio networks. In *Proc. of the IEEE Int. Conf. on Evolutionary Computation*, pages 539 – 544, 1995.

[74] R. Dorne and J.-K. Hao. Constraint handling in evolutionary search: a case study of the frequency assignment. In *Parallel Problem Solving from Nature (PPSN I)*, pages 801 – 810, 1996.

[75] B. Dorronsoro. *Diseño e implementación de algoritmos genéticos celulares para problemas complejos*. PhD thesis, University of Málaga, 2007.

[76] A. Durresi, V. K. Paruchuri, S. S. Iyengar, and R. Kannan. Optimized broadcast protocol for sensor networks. *IEEE Transactions on Computers*, 58(8):1013 – 1024, 2005.

[77] F. Y. Edgeworth. *Mathematical Psychics*. P. Keagan, London, 1881.

[78] M. Ehrgott. *Multicriteria Optimization*. Springer, second edition, 2005.

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

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

[81] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetic Algorithms 1*, pages 265 – 283. Morgan Kaufmann, 1991.

[82] FAP Web. http://fap.zib.de/.

[83] E. Fasolo, A. Zanella, and M. Zorzi. An effective broadcast scheme for alert message propagation in vehicular ad hoc networks. In *IEEE International Conference on Communications (ICC 2006)*, pages 3960 – 3965, 2006.

[84] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109 – 133, 1999.

[85] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14 – 19, 1962.

[86] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proc. of the Fifth Int. Conference on Genetic Algorithms*, pages 416 – 423, 1993.

[87] I. Foster. What is the grid? a three point checklist. *Grid Today*, 1(6), 2002.

[88] I. Foster. Globus toolkit version 4: Software for service-oriented system. *Journal of Computer Science and Technology*, 21(4):513 – 520, 2006.

[89] I. Foster and C. Kesselman. Globus: a metacomputing infraestructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115 – 128, 1997.

[90] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann, 1999.

[91] A. S. Fraser. Simulation of genetic systems by automatic digital computers II: Effects of linkage on rates under selection. *Australian Journal of Biological Sciences*, 10:492 – 499, 1957.

[92] A. Furuskar, J.Ñaslund, and H. Olofsson. EDGE – enhanced data rates for GSM and TD-MA/136 evolution. *Ericsson Review*, (1), 1999.

[93] M. Galota, C. Glasser, S. Reith, and H. Vollmer. A polynomial-time approximation scheme for base station positioning in UMTS networks. In *5th Discrete Algorithms and Methods for Mobile Computing and Communications Conference*, pages 52 – 59, 2000.

[94] N. Gerlich, K. Tutschku, and P. Tran-Gia. An integrated approach to cellular network planning. In *7th International Telecommunication Network Planning Symposium*, 1996.

[95] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156 – 166, 1977.

[96] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533 – 549, 1986.

[97] F. Glover. A template for Scatter Search and Path Relinking. In J-K. Hao et al., editor, *Artificial Evolution*, number 1363 in LNCS, pages 13–54. Springer, 1998.

[98] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653 – 684, 2000.

[99] F. Glover, M. Laguna, and R. Martí. Scatter search. In *Advances in Evolutionary Computing: Theory and Applications*, pages 519 – 537. Springer, 2003.

[100] F. W. Glover and G. A. Kochenberger. *Handbook of Metaheuristics*. Kluwer, 2003.

[101] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[102] R. L. Graham. Bounds on multiprocessor timing anomalies. *SIAM Journal of Applied Mathematics*, 17:416 – 429, 1969.

[103] H. Granbohm and J. Wiklund. GPRS – general packet radio service. *Ericsson Review*, (1), 1999.

[104] J.-Y. Greff, L. Idoumghar, and R. Schott. Application of markov decision processes to the frequency assignment problem. *Applied Artificial Intelligence*, 18(8):761 – 773, 2004.

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

[106] J. K. Han, B. S. Park, Y. S. Choi, and H. K. Park. Genetic approach with a new representation for base station placement in mobile communications. In *IEEE 54th Vehicular Technology Conference*, pages 2703 – 2707, 2001.

[107] J-K. Hao and R. Dorne. Study of genetic search for the frequency assignment problem. In *Artificial Evolution. European Conference, AE 95. Selected-Papers*, pages 333 – 344, 1996.

[108] J-K. Hao and L. Perrier. Tabu search for the frequency assignment problem in cellular radio networks. Technical Report LGI2P, EMA-EERIE, Parc Scientifique Georges Besse, 1999.

[109] Q. Hao, B. Soong, E. Gunawan, J. Ong, C. B. Soh, and Z. Li. A hierarchical optimization network resource planning approach. *IEEE Journal on Selected Areas in Communications*, 15(7):1315 – 1326, 1997.

[110] F. Herrera, M. Lozano, and D. Molina. Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. *European Journal of Operational Research*, 169:450 – 476, 2006.

[111] Y. Hochberg and A. C. Tamhane. *Multiple Comparison Procedures*. Wiley, 1987.

[112] L. Hogie, M. Seredynski, F. Guinand, and P. Bouvry. A bandwidth-efficient broadcasting protocol for mobile multi-hop ad hoc networks. In *5th International Conference on Networking (ICN'06)*, page 71. IEEE Press, 2006.

[113] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[114] J.H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297 – 314, 1962.

[115] H. Holma and A. Toskala. *WCDMA for UMTS*. Wiley, 2000.

[116] X. Huang, U. Berh, and W. Wiesbeck. Automatic base station placement and dimensioning for mobile network planning. In *IEEE Vehicular Technology Conference*, pages 1544 – 1549, 2000.

[117] X. Huang, U. Berh, and W. Wiesbeck. Automatic cell planning for a low-cost and spectrum efficient wireless network. In *Proceedings of Global Telecommunications Conference (GLOBECOM)*, pages 276 – 282, 2000.

[118] X. Huang, U. Berh, and W. Wiesbeck. A new approach to automatic base station placement in mobile networks. In *International Zurich Seminar on Broadband Communications*, pages 301 – 306, 2000.

[119] S. Hurley. Planning effective cellular mobile radio networks. *IEEE Transactions on Vehicular Technologies*, 51(2):243 – 253, 2002.

[120] S. Hurley and D. H. Smith. Fixed spectrum frequency assignment using natural algorithms. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, pages 373 – 378, 1995.

[121] T. S. Hussain. An introduction to evolutionary computation. tutorial presentation. CITO Researcher Retreat, May 12-14, Hamilton, Ontario 1998.

[122] L. Idoumghar and R. Schott. A new hybrid GA-MDP algorithm for the frequency assignment problem. In *Proc. of the 18th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI'06)*, pages 18 – 25, 2006.

[123] ITU. Propagation by diffraction. Technical Report UIT-R P.526-5, International Telecommunication Union, 1997.

[124] F. J. Jaimes-Romero, D. Muñoz-Rodríguez, and S. Tekinay. Channel assignment in cellular systems using genetic algorithms. In *Vehicular Technology Conference*, pages 741 – 745, 1996.

[125] N. Jaldén. Autonomous frequency planning for GSM networks. Master's thesis, Royal Institute of Technology, Stockholm, 2004.

[126] S. Ben Jamaa, Z. Altman, J. M. Picard, and B. Fourestié. Combined coverage and capacity optimisation for umts networks. In *11th International Telecommunications Network Strategy and Planning Symposium*, pages 175 – 178, 2004.

[127] S. Ben Jamaa, Z. Altman, J. M. Picard, and B. Fourestié. Multi-objective strategies for automatic cell planning of UMTS networks. In *2004 IEEE 59th Vehicular Technology Conference*, pages 2420 – 2424, 2004.

[128] B. Jaumard, O. Marcotte, and C. Meyer. *Telecommunications Network Planning*, chapter Mathematical models and exact methods for channel assignment in cellular networks, pages 239 – 256. Kluwer, 1999.

[129] A. Jedidi, A. Caminada, and G. Finke. 2-objective optimization of cells overlap and geometry with evolutionary algorithms. In *EvoCOMNET 2004*, LNCS 3005, pages 130 – 139, 2004.

[130] H. G. Gauch Jr. *Scientific Method in Practice*. Cambridge University Press, 2002.

[131] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. *Communications of the ACM*, 33(5):539 – 543, 1990.

[132] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Mathematics of Operations Research*, 2:209 – 224, 1977.

[133] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC 1999)*, pages 1931 – 1938, 1999.

[134] S-S. Kim, A. E. Smith, and J-H. Lee. A memetic algorithm for channel assignment in wireless FDMA systems. *Computers & Operations Research*, 34:1842 – 1856, 2007.

[135] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671 – 680, 1983.

[136] J. Knowles and D. Corne. The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 9 – 105. IEEE Press, 1999.

[137] J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report TIK-Report 214, Computer Engineering and Networks Laboratory, ETHC Zurich, 2006.

[138] 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.

[139] J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

[140] A. G. Krishna and K. M. Rao. Multi-objective optimisation of surface grinding operations using scatter search approach. *International Journal of Advanced Manufacturing Technology*, 29(5):475 – 480, 2006.

[141] W. Kuś. Grid-enabled evolutionary algorithm application in the mechanical optimization problems. *Engineering Applications of Artificial Intelligence*, 20(5):629 – 636, 2007.

[142] A. M. J. Kuurne. On GSM mobile measurement based interference matrix generation. In *IEEE 55th Vehicular Technology Conference, VTC Spring 2002*, pages 1965 – 1969, 2002.

[143] Manuel Laguna and Rafael Martí. *Scatter Search. Methodology and Implementations in C*. Kluwer, 2003.

[144] I. Laki, L. Farkas, and L.Ñagy. Cell planning in mobile communication systems using sga optimization. In *EUROCON'2001*, pages 124 – 127, 2001.

[145] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña. Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report KZZA-IK-4-99, Department of Computer Science and Artificial Intelligence, University of the Basque Country, 1999.

[146] T. L. Lau and E. P. K. Tsang. Guided genetic algorithm and its application to radio link frequency assignment problems. *Constraints*, 6(4):373 – 398, 2001.

[147] C. Y. Lee and H. G. Kang. Cell planning with capacity expansion in mobile communications: A tabu search approach. *IEEE Transactions on Vehicular Technology*, 49(5):1678 – 1691, 2000.

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

[149] G. Leguizamón and Z. Michalewicz. A new version of Ant System for subset problems. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1459 – 1464. IEEE Computer Society Press, 1999.

[150] K. Lieska, E. Laitinen, and J. Lähteenmäki. Radio coverage optimization with genetic algorithms. In *Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 318 – 322, 1998.

[151] 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.

[152] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 61 – 68, 2000.

[153] H.-P. Lin, R.-T. Juang, D.-B. Lin, C.-Y. Ke, and Y. Wang. Cell planning scheme for wcdma systems using genetic algorithm and measured background noise floor. *IEE Proceedings Communications*, 151(6):595 – 600, 2004.

[154] J. Linderoth, S. Kulkarni, J. P. Goux, and M. Yoder. An enabling framework for master-worker applications on the computational grid. In *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC)*, pages 43 – 50, 2000.

[155] J-P. M. G. Linmartz. *Wireless Communication.* Baltzer Science Publisher, 1996.

[156] H. R. Lourenço, O. Martin, and T. Stützle. *Handbook of Metaheuristics*, chapter Iterated local search, pages 321 – 353. Kluwer Academic Publishers, 2002.

[157] F. Luna, E. Alba, and A. J. Nebro. Parallel heterogeneous metaheuristics. In E. Alba, editor, *Parallel Metaheuristics*, pages 395 – 422. Wiley, 2005.

[158] F. Luna, E. Alba, A. J. Nebro, P. Mauroy, and S. Pedraza. Applying evolutionary algorithms to solve the automatic frequency planning problem. In A. Nayak and I. Stojmenovic, editors, *Handbook of Applied Algorithms: Solving Scientific, Engineering and Practical Problems.* Wiley (to appear), 2008.

[159] F. Luna, C. Blum, E. Alba, and A. J. Nebro. ACO vs EAs for solving a real-world frequency assignment problem in GSM networks. In *Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 94 – 101, 2007.

[160] F. Luna, B. Dorronsoro, A. J. Nebro, E. Alba, and P. Bouvry. Multiobjective metaheuristics to optimize the broadcasting in MANETs. In L.T. Yang and M. Denko, editors, *Handbook on Mobile Ad Hoc and Pervasive Communications.* American Scientific Publishers (to appear), 2008.

[161] F. Luna, C. Estébanez, C. León, J. M. Chávez González, E. Alba, R. Aler, C. Segura, M. A. Vega Rodríguez, A. J. Nebro, J. M. Valls, G. Miranda, and J. A. Gómez Pulido. Metaheuristics for solving a real-world frequency assignment problem in gsm networks. In *Genetic and Evolutionary Computation Conference - GECCO 2008 (to appear)*, 2008.

[162] F. Luna, A. J. Nebro, and E. Alba. Observations in using grid-enabled technologies for solving multi-objective optimization problems. *Parallel Computing*, 32(5-6):377 – 393, June 2006.

[163] F. Luna, A. J. Nebro, and E. Alba. Parallel evolutionary multiobjective optimization. In N. Nedjah, E. Alba, and L. de Macedo, editors, *Parallel Evolutionary Computations*, volume 22 of *Studies in Computational Intelligence*, chapter 2, pages 33 – 56. Springer, 2006.

[164] F. Luna, A. J. Nebro, E. Alba, and J. J. Durillo. Solving large-scale real-world telecommunication problems using a grid-based genetic algorithm. *Engineering Optimization (en segunda revisión con cambios menores)*, 2008.

[165] G. Luque. *Resolución de Problemas Combinatorios con Aplicación Real en Sistemas Distribuidos*. PhD thesis, University of Málaga, 2006.

[166] P. Lyster, L. Bergman, P. Li, D. Stanfill, B. Crippe, R. Blom, and D. Okaya. CASA gigabit supercomputing network: Calcrust three-dimensional real-time mult-dataset rendering. In *Proc. Supercomputing'92*, 1992.

[167] H. Mabed, A. Caminada, J.-K. Hao, , and D. Renaud. A dynamic traffic model for frequency assignment. In *Paralle Problem Solving from Nature (PPSN VII)*, LNCS 2439, pages 779 – 788, 2002.

[168] V. Maniezzo and A. Carbonaro. An ANTS heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 16(9):927 – 935, 2000.

[169] C. Maple, L. Guo, and J. Zhang. Parallel genetic algorithms for third generation mobile network planning. In *International Conference on Parallel Computing in Electrical Engineering*, pages 229 – 236, 2004.

[170] R. Martí, H. Loureno, and M. Laguna. Assigning Proctors to Exams with Scatter Search. In M. Laguna and J. L. González-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 215 – 227. Kluwer Academic Publishers, 2000.

[171] S. Matsui, I. Watanabe, and K-I. Tokoro. An efficient hybrid genetic algorithm for a fixed channel assignment problem with limited bandwidth. In *Genetic and Evolutionary Computation Conference (GECCO 2003)*, LNCS 2724, pages 2240 – 2251, 2003.

[172] S. Matsui, I. Watanabe, and K-I. Tokoro. Application of the parameter-free genetic algorithm to the fixed channel assignment problem. *Systems and Computers in Japan*, 36(4):71 – 81, 2005.

[173] N. Melab, S. Cahon, and E-G. Talbi. Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing*, 66:1052 – 1061, 2006.

[174] G. Mendel. *Versuche über Pflanzen-Hybriden*. Verhandlungen des Naturforschedes Vereines in Brünn 4, 1865.

[175] 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.

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

[177] H. Meunier, E-G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 317 – 324, 2000.

[178] A. R. Mishra. *Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G*. Wiley, 2004.

[179] A. R. Mishra. *Fundamentals of Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G*, chapter Radio Network Planning and Optimisation, pages 21 – 54. Wiley, 2004.

[180] N. Mladenovic and P. Hansen. Variable neighborhood search. *Com. Oper. Res*, 24:1097 – 1100, 1997.

[181] A. Molina, G. E. Athanasiadou, and A. R. Nix. The automatic location of base-stations for optimised cellular coverage: a new combinatorial approach. In *1999 IEEE 49th Vehicular Technology Conference*, pages 606 – 610, 1999.

[182] J. Molina, M. Laguna, R. Martí, and R. Caballero. SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization. *INFORMS Journal on Computing*, 19(1):91 – 100, 2007.

[183] D. Montana and J. Redi. Optimizing parameters of a mobile ad hoc network protocol with a genetic algorithm. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 1993 – 1998, 2005.

[184] R. Montemanni, D. H. Smith, and S. M. Allen. An ANTS algorithm for the minimum-span frequency assignment problem with multiple interference. *IEEE Transactions on Vehicular Technology*, 51(5):949 – 953, 2002.

[185] J.Ñ. J. Moon, L. A. Hughes, and D. H. Smith. Assignment of frequency lists in frequency hopping networks. *IEEE Trans. on Vehicular Technology*, 54(3):1147 – 1159, 2005.

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

[187] H. Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5:303 – 346, 1998.

[188] A. J. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, and J. J. Durillo. Optimal antenna placement using a new multi-objective CHC algorithm. In *Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 876 – 883, 2007.

[189] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. In D. A. Pelta and N. Krasnogor, editors, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2006)*, pages 25 – 36, 2006.

[190] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems (to appear)*, 2007.

[191] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. Design issues in a multiobjective cellular genetic algorithm. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *Evolutionary Multi-Criterion Optimization. 4th International Conference, EMO 2007*, volume 4403 of *Lecture Notes in Computer Science*, pages 126 – 140. Springer, 2007.

[192] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham. AbYSS: Adapting scatter search to multiobjective optimization. *IEEE Transactions on Evolutionary Computation. Accepted for publication.*, 2008.

[193] A. J. Nebro, G. Luque, F. Luna, and E. Alba. DNA fragment assembly using a grid based genetic algorithm. *Computers & Operations Research (to appear)*, 2007.

[194] S-Y. Ni, Y-C. Tseng, Y-S. Chen, and J-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. of the 5th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking*, pages 151 – 162, 1999.

[195] OR Library. http://people.brunel.ac.uk/∼mastjjb/jeb/info.html.

[196] A. Osyczka. Multicriteria optimization for engineering design. In J. S. Gero, editor, *Design Optimization*, pages 193 – 227. Academic Press, 1895.

[197] V. Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[198] B. S. Park, J. G. Yook, and H. K. Park. The determination of base station placement and transmit power in an inhomogeneous traffic distribution for radio network planning. In *2002 IEEE 56th Vehicular Technology Conference Proceedings*, pages 2051 – 2055, 2002.

[199] A. Pelc. *Handbook of Wireless Networks and Mobile Computing*, chapter Broadcasting In Wireless Networks, pages 509 – 528. Wiley, 2002.

[200] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, volume 1, pages 525 – 532. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[201] W. Peng and X-C. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *MobiHoc '00: Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 129 – 130. IEEE Press, 2000.

[202] J. M. Picard, Z. Altman, S. Ben Jamaa, M. Demars, H. Dubreil, B. Fourestié, and A. Ortega. Automatic cell planing strategies for umts networks. *Int. J. of Mobile Network Design and Innovation*, 1(1):8 – 17, 2005.

[203] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Unive. Press, 1992.

[204] A. R. Rahimi-Vahed, M. Rabbani, R. Tavakkoli-Moghaddam, S. A. Torabi, and F. Jolai. A multi-objective scatter search for a mixed-model assembly line sequencing problem. *Advanced Engineering Informatics*, 21(1):85 – 99, 2007.

[205] L. Raisanen. A permutation-coded evolutionary strategy for multi-objective GSM network planning. *Journal of Heuristics. Forthcoming.*, 2008.

[206] L. Raisanen and R. M. Whitaker. Multi-objective optimization in area coverage problems for cellular communication networks: Evaluation of an elitist evolutionary strategy. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 714 – 720, 2003.

[207] L. Raisanen and R. M. Whitaker. Comparison and evaluation of multiple objective genetic algorithms for the antenna placement problem. *MONET*, 10(1-2):79 – 88, 2005.

[208] L. Raisanen, R. M. Whitaker, and S. Hurley. A comparison of randomized and evolutionary approaches for optimizing base station site selection. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 1159 – 1165, 2004.

[209] A. Rama Mohan Rao and N. Arvind. A scatter search algorithm for stacking sequence optimisation of laminate composites. *Composite Structures*, 70(4):383 – 402, October 2005.

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

[211] A. Raymond, V. Lyandres, and R. Chávez Santiago. On a guided genetic algorithm for frequency assignment in non-homogeneous cellular networks. In *2003 IEEE International Symposium on Electromagnetic Compatibility (EMC '03)*, pages 660 – 663, 2003.

[212] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., UK, 1965.

[213] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.

[214] C.R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, Oxford, UK, 1993.

[215] P. Reininger and A. Caminada. Connectivity management on mobile network design. In *10th Conf. Eur. Consortium for Mathematics in Industry*, 1998.

[216] P. Reininger and A. Caminada. Model for GSM radio network optimization. In *Second ACM International Conference on Discrete Algorithms and Methods for Mobility*, 1998.

[217] P. Reininger and A. Caminada. Multicriteria design model for cellular network. *Annals of Operations Research*, 107:251 – 265, 2001.

[218] P. Reininger, S. Iksal, A. Caminada, and J. J. Korczak. Multi-stage optimization for mobile radio network planning. In *1999 IEEE 49th Vehicular Technology Conference*, pages 2034 – 2038, 1999.

[219] M. G. C. Resende and P. M. Pardalos, editors. *Handbook of Optimization in Telecommunications*. Springer, 2006.

[220] A. Rogers and A. Prügel-Bennett. Genetic drift in genetic algorithm selection schemes. *IEEE Transactions on Evolutionary Computation*, 3(4):298 – 303, 1999.

[221] A. Rogers and A. Prügel-Bennett. Modelling the dynamics of a steady-state genetic algorithm. In W. Banzhaf and C. Reeves, editors, *Foundations of Genetic Algorithms 5*, pages 57 – 68. Morgan Kaufmann, San Francisco, CA, 1999.

[222] S. Ruíz, X. Colet, and J. J. Estevez. Frequency planning optimisation in real mobile networks. In *IEEE VTS 50th Vehicular Technology Conference*, pages 2082 – 2086, 1999.

[223] S. Salcedo-Sanz and C. Bousoño-Calzón. A portable and scalable algorithm for a class of constrained combinatorial optimization problems. *Computers & Operations Research*, 32:2671 – 2687, 2005.

[224] H.-P. Schwefel. *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik*. PhD thesis, Technical University of Berlin, 1965.

[225] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. CRC Press, 2003.

[226] M. K. Simon and M-S. Alouini. *Digital Communication over Fading Channels: A Unified Approach to Performance Analysis*. Wiley, 2005.

[227] L. Smarr and C. Catlett. Metacomputing. *Communications of the ACM*, 35(6):44 – 52, 1992.

[228] K. Socha and C. Blum. *Metaheuristic Procedures for Training Neural Networks*, chapter 8, Ant Colony Optimization, pages 153 – 180. Springer, 2006.

[229] M. Soto, A. Ochoa, S. Acid, and L. M. de Campos. Introducing the polytree aproximation of distribution algorithm. In *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99*, pages 360 – 367, 1999.

[230] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221 – 248, 1994.

[231] H. Stockinger. Defining the grid: a snapshot on the current view. *Journal of Supercomputing*, 42:3 – 17, 2007.

[232] I. Stojmenovic and J. Wu. *Broadcasting and activity-scheduling in ad hoc networks*, pages 205 – 229. IEEE/Wiley, 2004.

[233] R. Storn and K. Price. Differential evolution – a simple efficient adaptive scheme for global optimization over continuous spaces. Technical Report 95-012, Int. Compt. Sci. Inst., Berkeley, CA, 1995.

[234] T. Stützle. Local search algorithms for combinatorial problems analysis, algorithms and new applications. Technical report, DISKI Dissertationen zur Künstliken Intelligenz. Sankt Augustin, Germany, 1999.

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

[236] E-G. Talbi, S. Cahon, and N. Melab. Designing cellular networks using a parallel hybrid metaheuristic on the computational grid. *Computer Communications*, 30:698 – 713, 2007.

[237] E-G. Talbi and H. Meunier. Hierarchical parallel approach for gsm mobile network design. *Journal of Parallel and Distributed Computing*, 66(2):274–290, 2006.

[238] D. Thain, T. Tannenbaum, and M. Livny. Condor and the grid. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 299 – 335. John Wiley & Sons Inc., 2003.

[239] K. Tutschku. Interference minimization using automatic design of cellular communication networks. In *IEEE Vehicular Technology Conference*, pages 634 – 638, 1998.

[240] C. Valenzuela. A study of permutation operators for minimum span frequency assignment using an order based representation. *Journal of Heuristics*, 7:5 – 21, 2001.

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

[242] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Wright-Patterson, AFB, OH, 1999.

[243] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., Wright-Patterson, AFB, OH, 1998.

[244] J. A. Vasconcelos, J. H. R. D. Maciel, and R. O. Parreiras. Scatter search techniques applied to electromagnetic problems. *IEEE Transactions on Magnetics*, 4(5):1804 – 1807, 2005.

[245] M. Vasquez and J-K. Hao. A heuristic approach for antenna positioning in cellular networks. *Journal of Heuristics*, 7:443 – 472, 2001.

[246] M. A. Vega-Rodríguez, J. A. Gómez-Pulido, E. Alba, D. Vega-Pérez, S. Priem-Mendes, and G. Molina. Evaluation of different metaheuristics solving the RND problem. In *EvoCOMNET 2007*, LNCS 4448, pages 101 – 110, 2007.

[247] M. A. Vega-Rodríguez, D. Vega-Pérez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez. Radio network design using population-based incremental learning and grid computing with BOINC. In *EvoCOMNET 2007*, LNCS 4448, pages 91 – 100, 2007.

[248] C. Voudouris and E. Tsang. Guided local search. *European Journal of Operations Research*, 113(2):449 – 499, 1999.

[249] B. H. Walke. *Mobile Radio Networks: Networking, protocols and traffic performance*. Wiley, 2002.

[250] N. Weicker, G. Szabo, K. Weicker, and P. Widmayer. Evolutionary multiobjective optimization for base station transmitter placement with frequency assignment. *IEEE Transactions on Evolutionary Computation*, 7(2):189 – 203, 2003.

[251] B. Weinberg, V. Bachelet, and E-G. Talbi. A co-evolutionist meta-heuristic for the assignment of the frequencies in cellular networks. In *EvoWorkshop 2001*, LNCS 2037, pages 140 – 149, 2001.

[252] W. Whewell and R. E. Butts. *Theory of Scientific Method*. Hackett Pub Co Inc, 1989.

[253] R. Whitaker, L. Raisanen, and S. Hurley. Explicit characterization of tensions in downlink cell planning. In *2004 IEEE 60th Vehicular Technology Conference*, pages 3434 – 3437, 2004.

[254] R. Whitaker, L. Raisanen, and S. Hurley. A model for conflict resolution between coverage and cost in cellular wireless networks. In *Proceedings of the 37th Hawaii International Conference on System Sciences*, page 90287a, 2004.

[255] J. Whittaker. *Graphical models in applied multivariate statistics*. John Wiley & Sons, Inc., 1990.

[256] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proc. of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 194 – 205, 2002.

[257] J. Wu and W. Lou. Forward-node-set-based broadcast in clustered mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 3(2):155 – 173, 2003.

[258] Y. Zhang, C. Ji, P. Yuan, M. Li, C. Wang, and G. Wang. Particle swarm optimization for base station placement in mobile communication. In *2004 IEEE International Conference on Networking, Sensing and Control*, pages 428 – 432, 2004.

[259] J. Zimmermann, R. Höns, and H. Mühlenbein. The antenna placement problem for mobile radio networks: an evolutionary approach. In *Proceedings of the 8th Int. Conference on Telecommunication Systems*, pages 358 – 366, 2000.

[260] J. Zimmermann, R. Höns, and H. Mühlenbein. *Advances in evolutionary computing: theory and applications*, chapter From theory to practice: an evolutionary algorithm for the antenna placement problem, pages 713 – 737. Springer, 2003.

[261] J. Zimmermann, R. Höns, and H. Mühlenbein. ENCON: evolutionary algorithm for the antenna placement problem. *Computers and Industrial Engineering*, 44:209 – 226, 2003.

[262] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[263] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257 – 271, 1999.

[264] E. Zitzler, L. Thiele, M. Laummans, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):114 – 132, 2003.

Índice de Tablas

Índice de figuras

Índice alfabético