INDICE

Introducción

En una red de comunicaciones ad-hoc los distintos nodos, normalmente inalámbricos, pueden situarse en cualquier punto del espacio. En estos casos, deben definirse los enlaces sobre la marcha, de tal forma que haya conectividad entre todos los nodos de la red. Una solución sencilla consistiría en cubrir desde cada nodo los restantes nodos de la red, es decir, en crear una red completa en que todos los nodos se comunicaran directamente entre sí a través de enlaces inalámbricos. En este contexto, sin embargo, un nodo tendría que alcanzar a los nodos más lejanos de la red, lo cual supondría un consumo considerable de energía. Este punto reviste especial importancia en las redes ad-hoc, cuyos nodos son normalmente elementos portátiles que funcionan con batería. De este modo, en las redes ad-hoc se intenta reducir en la medida de lo posible el radio de alcance de los distintos nodos, con el objetivo de minimizar el consumo de energía. Es decir, un nodo sólo alcanzará generalmente a los vecinos más próximos, de forma que para mandar información a los nodos más alejados lo hará a través de una serie de nodos intermedios.

El presente proyecto tiene como fin el diseño de la topología de una red ad-hoc con un doble requerimiento. De un lado, conseguir conectividad completa, es decir, garantizar que un nodo pueda mandar información a todos los nodos de la red, aunque sea a través de una serie de nodos intermedios. Y, en segundo lugar, lograr que el consumo total de energía sea mínimo. Puesto que tenemos redes inalámbricas, consideraremos que el consumo de energía es proporcional al cuadrado de su radio de cobertura, de modo que finalmente trataremos de minimizar la suma de los cuadrados del radio de cobertura de los nodos de la red.

El problema planteado –que en términos teóricos se denomina subgrafo de difusión de energía mínima– es un problema NP-completo, es decir, un problema que no

puede resolverse en tiempos breves de computación. Es por ello que en el presente proyecto se han utilizado dos algoritmos probabilistas –simulated annealing y algoritmo hormigas– capaces de obtener soluciones cercanas a la solución óptima en tiempos razonables de cálculo. Ambos algoritmos se han comparado, además, con un algoritmo determinista basado en el algoritmo Kruskal.

La memoria se divide de la siguiente manera. En el primer capítulo se describen las principales características y aplicaciones de las redes ad-hoc, y se recuerda la importancia del consumo de las baterías en el diseño de la topología de la red. En el segundo capítulo se presentan las herramientas teóricas para la definición formal del problema de la asignación de los radios de cobertura en una red ad-hoc. Como en la mayoría de problemas que plantean las redes de comunicación, dicho marco teórico se conoce con el nombre de teoría de grafos. A continuación, en el tercer capítulo, se presentan las herramientas propuestas para la resolución aproximada del problema y en el cuarto capítulo se especifican con más detalle los procedimientos aplicados en el contexto particular del presente proyecto. Finalmente, en el capítulo quinto se muestran los resultados obtenidos con los distintos procedimientos planteados, se presentan las conclusiones y se describen las principales líneas futuras de investigación.

1. Redes ad-hoc

El éxito de las comunicaciones inalámbricas y la progresiva reducción en el tamaño de los dispositivos con capacidad para comunicaciones de datos ha multiplicado en los últimos cinco años el estudio de las redes móviles ad-hoc. En este capítulo se presentan las características generales de estas redes, que constituyen el marco de trabajo del presente proyecto.

1.1 *Breve historia*

La locución latina ad-hoc significa literalmente "para esto", es decir, "con un propósito determinado". Según esta definición, una red ad-hoc se constituiría con un cierto propósito gracias a un conjunto de dispositivos independientes que pueden establecer enlaces inalámbricos entre sí y conformar una red en ausencia de una infraestructura previa. La Figura 1.1 muestra un ejemplo de red ad-hoc.

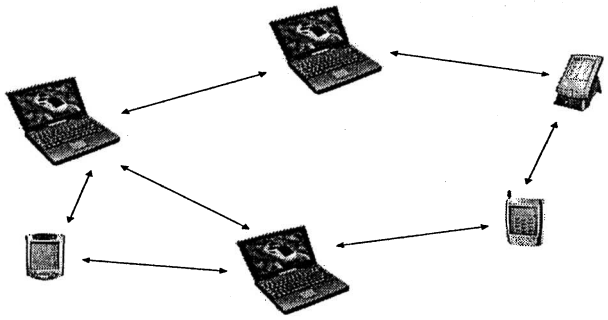

Figura 1.1.- Red ad-hoc

Las redes ad-hoc tienen un origen militar. De forma casi contemporánea al nacimiento de Internet, a principios de los 70, el ministerio de defensa americano se interesó por un proyecto conocido como Packet Radio Networks (PRNETs). Su

objetivo era comunicar las distintas unidades de un campo de batalla mediante dispositivos de radio, con libertad de movimiento y de forma que cada nodo pudiera comportarse no sólo como un elemento receptor o transmisor de información, sino también como un nodo intermedio de la red, en cuyo caso tendría que servir de puente de comunicación entre otros puntos de la red.

Durante los años posteriores a su aparición, la investigación en redes ad-hoc se circunscribió únicamente al campo militar. En los últimos años, no obstante, tanto la amplia difusión de las comunicaciones inalámbricas como la progresiva reducción del tamaño de los dispositivos de comunicación, ha despertado el interés por las redes ad-hoc y por las numerosas aplicaciones civiles que pueden derivarse de su uso.

1.2 *Características*

Como se ha dicho, los nodos de una red ad-hoc son normalmente dispositivos móviles, lo cual no excluye la presencia de nodos fijos, como por ejemplo un ordenador personal de sobremesa. En cualquier caso, los dispositivos que forman una red ad-hoc, por definición, pueden cambiar de posición libremente y comunicarse entre sí a través de enlaces inalámbricos.

De este modo, la topología de una red ad-hoc es variable, de forma que un nodo que dispone de un enlace con un nodo vecino puede desplazarse, desaparecer de su radio de cobertura y formar un nuevo enlace con un tercer nodo visible en su nueva zona de cobertura. La desaparición de enlaces como consecuencia de la movilidad de los nodos se traduce en una variación de las rutas de encaminamiento mucho mayor que en las redes clásicas de comunicaciones, incluida Internet. La Figura 1.2 ilustra esta situación.

Como se ha dicho, en una red ad-hoc los nodos se comunican a través de enlaces inalámbricos, que tienen un ancho de banda menor que los enlaces fijos y son además más propensos a los errores de transmisión. Por otro lado, dada la duración

limitada de las baterías de los nodos portátiles, estos transmiten a baja potencia, lo que se traduce en un bajo radio de alcance, que puede oscilar entre algunos metros y una longitud máxima de unos cientos de metros. Esta limitación de la cobertura de los nodos se trata de compensar mediante la colaboración, de modo que los nodos de una red ad-hoc no sólo emiten o reciben información propia, sino que además pueden actuar de repetidores entre dos nodos que carecen de visibilidad directa.

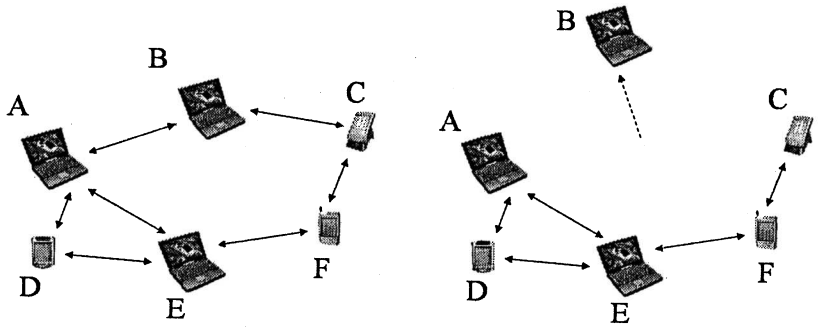

Figura 1.2.- El nodo B se desplaza y desaparece del área de cobertura de los nodos A y C. El tráfico de la ruta ABC deberá utilizar ahora otra ruta, como AEFC

Como se ha dicho, en las redes ad-hoc no se distingue entre dispositivos terminales y nodos intermedios, de forma que cualquier nodo puede desempeñar ambos papeles en cualquier momento, careciendo la red de una estructura jerárquica o centralizada. En este contexto, la arquitectura clásica de cliente-servidor en la que un nodo cliente pide un servicio a otro nodo servidor, deja además de tener sentido dado que todos los nodos pueden hacer a la vez de clientes y servidores.

Por otra parte, cabe considerar que los dispositivos de una red ad-hoc pueden tener una autonomía limitada, debido a su reducida capacidad de batería. Por ello, debe tenerse en cuenta el consumo de energía de modo que los programas de control permitan pasar al modo de bajo consumo en caso de inactividad. Igualmente, deben evitarse las retransmisiones innecesarias, así como las colisiones en el canal de acceso, fenómeno que se produce cuando dos nodos ocupan un mismo canal simultáneamente e interfieren entre ellos. En este caso se produce un error doble de transmisión, con la subsiguiente pérdida de eficiencia energética, ya que será necesario retransmitir los mensajes y volver a invertir energía en ellos.

En estos casos, se recomienda también disminuir la frecuencia de los mensajes de control, e incluso tener en cuenta el nivel de energía disponible en un dispositivo en el diseño de la topología de la red. En este contexto, podría eximirse a los nodos con problemas de energía de hacer de puentes entre otros nodos o, cuando menos, reducir el alcance de su radio de cobertura, todo con el objeto de aumentar su ciclo de vida.

1.3 *Protocolos en redes ad-hoc*

En los últimos veinte años la arquitectura de protocolos TCP/IP ha mostrado una enorme flexibilidad para adaptarse a una gran variedad de contextos tecnológicos, facultad que se ha traducido en la universalidad de la red basada en ambos protocolos: Internet. En rigor, Internet podría definirse como un conjunto de redes o como una red de redes que pueden tener diferentes plataformas tecnológicas pero que son capaces de intercambiar información mediante la familia de protocolos compartidos TCP/IP. Es por esto que las redes ad-hoc pueden definirse también sobre la base de los protocolos TCP/IP, hecho que además permitiría su sencilla integración en Internet. Sin embargo, ciertos escenarios ad-hoc pueden requerir protocolos adaptados a su naturaleza particular, en la que la topología puede ir cambiando a lo largo del tiempo.

Además, en las redes ad-hoc el protocolo TCP (que se emplea para proporcionar fiabilidad extremo a extremo, es decir, para asegurar que la información llega correctamente de un extremo a otro de la comunicación) no siempre funciona, por culpa de la escasa fiabilidad de las conexiones inalámbricas. TCP fue concebido para las comunicaciones por cable, que son generalmente más fiables, es decir, que provocan menos errores de transmisión y que también suelen proporcionar más ancho de banda, además de permanecer inamovibles a lo largo del tiempo. Es por ello que se han planteado varias propuestas de modificaciones para el uso de TCP en redes ad-hoc.

Dado que en las redes ad-hoc todos los nodos pueden desempeñar tanto funciones de cliente como funciones de servidor, y puesto que esos nodos pueden aparecer y desaparecer de la red dada su naturaleza portátil, debe existir algún mecanismo simple para determinar los nodos capaces de ofrecer un determinado servicio a otros. Por otra parte, también han de emplearse mecanismos de seguridad, tales como autentificación y privacidad, para prevenir el ataque de usuarios ajenos a la red pero que pueden acceder a ella invadiendo el espacio compartido a través de los enlaces inalámbricos.

1.4 *Encaminamiento en redes ad-hoc*

De los problemas que plantean las redes ad-hoc, el más complejo es el problema de encaminamiento, es decir, el de la búsqueda de caminos a lo largo de la red, para enviar información de un nodo a otro. Se trata de un problema especialmente delicado dado que cualquier nodo, como se ha dicho, no sólo puede hacer de puente entre otros nodos sino que puede además desplazarse de un punto a otro del espacio, deshaciendo rutas existentes pero permitiendo la formación de nuevos caminos. Los protocolos de encaminamiento de las redes ad-hoc han de tener, por tanto, una enorme capacidad de adaptación, de forma que puedan deshacer y rehacer rutas sobre la marcha sin una gran necesidad de cálculo, conforme los nodos se desplazan de un punto a otro de la red.

Los protocolos de encaminamiento en redes ad-hoc pueden dividirse en encaminamiento unicast, multicast y broadcast, según el destinatario de la información sea un único nodo, un conjunto de nodos o toda la red. En este proyecto nos hemos concentrado en el primer caso. De los protocolos de encaminamiento unicast pueden destacarse dos clases: los proactivos y los reactivos, además de los mecanismos híbridos.

En el encaminamiento proactivo las tablas de encaminamiento de los nodos están permanentemente actualizadas, de forma que cuando un nodo quiere enviar datos a otro, dispone siempre de la información necesaria para alcanzar su destino. En términos prácticos, esto supone un intercambio periódico de información entre los nodos, y por tanto, la existencia de un tráfico de control permanente en la red, lo que puede traducirse en una mayor probabilidad de congestión, de una parte, y en un menor tiempo de vida de las baterías, de otra. Sin embargo, la continua disponibilidad de rutas permite que los retardos sean mínimos, dado que un nodo tendrá disponible la ruta en el instante mismo de empezar la comunicación. Los protocolos preactivos más utilizados en redes ad-hoc son el Optimized Link State Routing protocol (OLSR) y el Topology dissemination Based on Reverse-Path Forwarding (TBRPF).

En oposición al funcionamiento del encaminamiento proactivo, los protocolos reactivos sólo buscan las rutas bajo demanda. De este modo, cuando un nodo quiere mandar un paquete a otro nodo, se pone en marcha un mecanismo para determinar la ruta que debe seguirse, reduciéndose de este modo el tráfico de control de la red, pero aumentando el retardo de la información ya que cada nodo, antes de empezar la transmisión, deberá esperar mientras se calcula la ruta de transmisión. Los protocolos de encaminamiento reactivos más empleados son el Ad-hoc On-demand Distance Vector routing (AODV) y el Dynamic Source Routing (DSR).

Las estrategias reactiva y proactiva representan dos estrategias extremas de encaminamiento en redes ad-hoc. Existen, no obstante, métodos que combinan ambas técnicas, el más popular de los cuales es el Zone Routing Protocol (ZRP). Este protocolo divide las redes ad-hoc en distintas regiones en las cuales realiza encaminamiento proactivo. Sin embargo, el protocolo funciona de forma reactiva entre zonas. Otra propuesta más sofisticada la ofrece el Sharp Hybrid Adaptive Routing Protocol (SHARP) que permite que el radio de las zonas proactivas aumente o disminuya teniendo en cuenta la movilidad de los nodos y el patrón de tráfico de la red. Esta medida puede tener un impacto significativo tanto en el retardo de la transmisión como en la tasa de pérdidas de los paquetes de datos.

1.5 *Aplicaciones*

Dependiendo de la naturaleza de sus nodos, pueden definirse dos clases de redes ad-hoc: las redes ad-hoc puras, que carecen de infraestructura, y las redes híbridas, que son más comunes y combinan una red ad-hoc con nodos que ofrecen acceso a otras redes como Internet.

Como se ha visto, la conectividad ad-hoc permite a los terminales de cualquier conjunto de usuarios construir una red instantáneamente cuando estos se concentran en un espacio compartido, como sucede en las conferencias o reuniones. En un caso así los distintos nodos pueden intercambiar información e incluso trabajar simultáneamente sin necesidad de una infraestructura fija. Las redes ad-hoc pueden servir también para calcular la posición de los distintos nodos de la red mediante medidas de triangularización, propiedad que puede resultar de gran utilidad en situaciones de emergencia, como por ejemplo cuando se pretende localizar a un miembro extraviado de un equipo de salvamento durante la extinción de un incendio.

Otra futura aplicación de las arquitecturas ad-hoc la constituyen las redes en ruta, esto es, redes en las que los vehículos funcionan como nodos de intercambio de información sobre el estado de las carreteras. De este modo, podría reducirse la

probabilidad de congestión desviando a los vehículos por las rutas menos transitadas en una determinada franja horaria. En un futuro, incluso, un vehículo podría prevenir a los vehículos posteriores de la existencia de un obstáculo en el terreno o de una mancha resbaladiza, con el objeto de disminuir el riesgo de sufrir un accidente. E incluso podrían evitarse las colisiones frontales puesto que un vehículo podría determinar la presencia de otro vehículo en el carril contrario antes de disponer de visibilidad. La arquitectura ad-hoc también reviste una especial importancia en las redes de sensores, en las que los distintos nodos pueden transmitirse instantáneamente sus medidas. En un caso así, la información podría mostrarse finalmente de forma centralizada y en tiempo real, si todas las medidas se dirigieran a una base de datos común.

Por otro lado, las redes híbridas disponen de nodos de infraestructura junto a nodos puramente ad-hoc. Dos ejemplos de uso de tales arquitecturas son las redes corporativas y las redes domésticas. Las redes corporativas suelen cubrir uno o varios edificios cercanos que forman parte normalmente de una misma empresa o campus universitario. En este escenario conviven los nodos de la infraestructura junto a dos tipos de nodos ad-hoc: nodos fijos alimentados por la red eléctrica y con las características de un PC de sobremesa, y nodos móviles de tamaño reducido y que se alimentan con una batería. Cualquiera de estos dispositivos ad-hoc puede actuar como puente entre otros nodos, pero los primeros deberían ser usados con mayor prioridad dado que no presentarán el problema del consumo de energía. Por otro lado, la propia infraestructura de la red proporcionaría acceso a Internet.

Por último, las redes domésticas tienen un claro interés en el campo de la domótica. En este caso los dispositivos de la red pueden conectarse a Internet, de modo que sean configurables de forma remota. Estos nodos podrían ser tanto ordenadores como electrodomésticos o elementos con algún tipo de control remoto. En un caso así, por ejemplo, podríamos activar remotamente la calefacción poco antes de llegar a casa o ser prevenidos en caso de que se produjera alguna alarma. E incluso podrían tomarse medidas estadísticas para optimizar el consumo de energía, tanto

en verano como en invierno, conectando el aire acondicionado o la calefacción según unas pautas óptimas de ahorro energético.

2. Teoría de grafos

En este capítulo se definen las herramientas teóricas necesarias para resolver el problema de la asignación de los radios de cobertura en una red ad-hoc. Dado que las redes se han representado finalmente mediante grafos, se describen a continuación los conceptos principales de dicha materia, así como el problema del árbol generador mínimo, que puede emplearse como método de resolución aproximada para el diseño de la topología de una red ad-hoc.

2.1 *Definiciones*

Un grafo simple G es un par $(V(G),E(G))$ tal que $V(G)$ es un conjunto finito de elementos denominados nodos o vértices y $E(G)$ un conjunto finito de pares no ordenados de nodos. Cada uno de los elementos de $E(G)$ se denomina arista o rama. Tanto los nodos como las aristas de un grafo pueden incluir etiquetas definidas a partir de aplicaciones $\Phi : V(G) \rightarrow R$ y $\Phi' : E(G) \rightarrow R$ de tal forma que cada nodo o arista tenga asociado un número. El orden n de un grafo $G=(V,E)$ es el número de nodos o cardinal de $V(G)$. Asimismo se define el tamaño E de un grafo $G=(V,E)$ como el cardinal de $E(G)$ o el número de aristas de G. Un grafo suele representarse gráficamente dibujando los nodos como puntos y las aristas como líneas que unen los nodos, según se observa en la Figura 2.1 y siguientes.

Según la definición, un grafo simple no puede tener aristas repetidas, esto es, pares de nodos unidos por más de una arista, ni tampoco lazos (aristas que enlazan un mismo nodo). En lo sucesivo, y a no ser que se especifique lo contrario, cada vez que nos refiramos a un grafo G se entenderá que G es un grafo simple.

Se dice que dos nodos u y v son adyacentes cuando los une una arista uv. En este caso se dice que los nodos u y v inciden sobre la arista uv o bien que la arista uv

incide sobre los nodos *u* y *v*. Dos aristas son adyacentes cuando tienen un nodo en común. El grado $\delta(v)$ de un nodo *v* es el número de aristas incidentes en *v*.

Dado un grafo simple de orden *n* el número máximo de aristas que puede contener es igual a *n(n-1)/2*, que son todas las combinaciones posibles que pueden formarse tomando los *n* nodos de dos en dos. Se define así la densidad $\rho(G)$ de un grafo *G* de orden *n* como el cociente entre el número de aristas del grafo *G* y el máximo número de aristas que puede contener un grafo de orden *n*. De este modo

$$\rho(G) = \frac{E}{n(n-1)/2} = \frac{2E}{n(n-1)}$$

Un clique $C \subset V(G)$ de un grafo *G* es un conjunto de nodos de *V(G)* tal que todo par de nodos de *C* son adyacentes entre sí. De otro lado se denomina conjunto independiente a un conjunto $S \subset V(G)$ de nodos tales que no son adyacentes dos a dos.

Una secuencia de aristas es una sucesión de aristas consecutivas v_0v_1, v_1v_2, v_2v_3, ..., $v_{m-1}v_m$. La secuencia dibuja un camino continuo sobre el grafo. Una secuencia en la que no se repiten aristas se denomina cola y si tampoco se repite ningún nodo, trayecto o camino. Un trayecto cerrado tal que el primer y el último nodo coinciden se denomina circuito o ciclo.

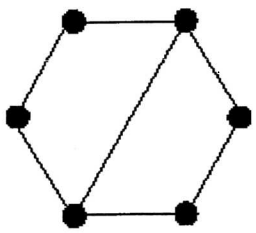

Figura 2.1.- Grafo conexo

Se dice que un grafo G es conexo cuando es posible trazar al menos un trayecto entre dos nodos cualesquiera de $V(G)$. Cada uno de los conjuntos conexos de nodos en que puede descomponerse un grafo no conexo se denomina componente conexa del grafo. La distancia $d(u,v)$ entre dos nodos u y v se define como el cardinal mínimo de todos los trayectos entre u y v, es decir, el número mínimo de aristas que hay que recorrer para llegar de un nodo a otro. Cuando un grafo no es conexo y dos nodos pertenecen a componentes distintas se dice que su distancia es infinita. Se denomina diámetro de un grafo G a la distancia máxima entre dos nodos cualesquiera de G.

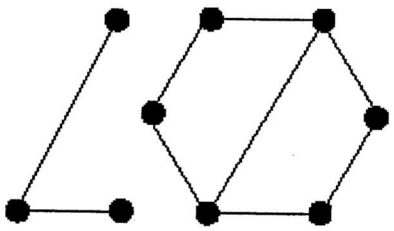

Figura 2.2.- Grafo no conexo

Un grafo completo K_n es un grafo simple tal que todos sus pares de nodos están unidos por una arista. De este modo el número de aristas de K_n es igual a $n(n-1)/2$. De otro lado, un grafo cuyos nodos tienen todos el mismo grado es un grafo regular. Concretamente se dice que G es un grafo regular de grado r si todos sus nodos tienen grado r.

Dados dos grafos cualesquiera $G_1=(V_1,E_1)$ y $G_2=(V_2,E_2)$, G_2 es subgrafo de G_1 si y sólo si $V_2 \subseteq V_1$ y $E_2 \subseteq E_1$. Si $V_2 = V_1$ entonces G_2 es subgrafo generador de G_1. En caso de que G_2 conserve las aristas de G_1, es decir, en el caso de que todo par de nodos del grafo G_2 sean adyacentes si son adyacentes en G_1, diremos que G_2 es un subgrafo inducido de G_1.

Dos grafos $G=(V,E)$ y $G'=(V',E')$ son isomorfos si existe una biyección $\Phi : V \rightarrow V'$ tal que para todo par de nodos $u,v \in V(G)$ la arista uv pertenece a $E(G)$ si y sólo si la arista $\Phi(u)\Phi(v)$ pertenece a $E'(G')$. En este caso se dice que Φ es un isomorfismo de G en G'. Dos grafos isomorfos G y G' pueden representarse gráficamente de la misma manera. Un automorfismo de un grafo G es un isomorfismo de G en G. Un grafo $G=(V,E)$ es vértice-transitivo o vértice-simétrico si dada una pareja arbitraria de nodos $u,v \in V(G)$ existe un automorfismo Φ de G tal que $\Phi(u) = v$. Dado un grafo vértice-transitivo todos los nodos son intercambiables y tienen las mismas propiedades.

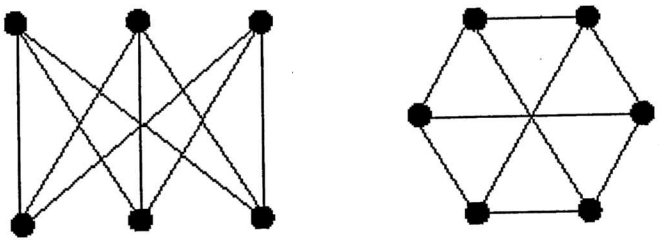

Figura 2.3.- Grafos isomorfos vértice-transitivos

Un grafo $G=(V,E)$ es plano si puede dibujarse sobre el plano sin que ninguna de sus aristas se corte. De otro lado un grafo $G=(V,E)$ es planar si existe algún grafo plano isomorfo a G.

Un grafo G es k-coloreable o k-partito si existe una aplicación $\Phi : V(G) \rightarrow \{1,\ldots,k\}$ tal que $\Phi(u) \neq \Phi(v)$ para todo par de nodos u y v adyacentes, es decir, tal que dos nodos adyacentes tengan siempre imágenes distintas. La aplicación Φ suele denominarse coloreado de los nodos del grafo G y cada uno de los enteros asignados a cada nodo recibe el nombre de color. El número cromático $X(G)$ de un grafo G es el mínimo entero k tal que G es k-coloreable. De este modo, el coloreado de un grafo

consiste en asignar un color (o etiqueta o número) a cada uno de los nodos del grafo de tal manera que dos nodos adyacentes tengan colores distintos. Y el número cromático de un grafo es el mínimo número de colores necesarios para colorear el grafo correctamente.

Un digrafo *G* se define como un par *(V(G),A(G))*, donde *V(G)* es un conjunto finito no vacío de elementos llamados nodos, y *A(G)* es una familia finita de pares ordenados de elementos de *V(G)* llamados arcos o aristas dirigidas o también, por simplicidad, aristas. *V(G)* y *A(G)* se denominan conjunto de nodos y familia de arcos de *G*, respectivamente. En las ocasiones en que no existe ambigüedad, puede emplearse el término grafo para designar a un digrafo.

De un arco cuyo primer elemento es *v* y cuyo segundo elemento es *w* se dice que es un arco de *v* a *w*, y es designado *(v,w)* o simplemente *vw*. Adviértase que los arcos *vw* y *wv* son diferentes. Si *G* no posee ningún lazo (arcos de la forma *vv*) y todos los arcos de *G* son diferentes, *G* es un digrafo simple. Los arcos de un digrafo *G* pueden incluir también etiquetas definidas por aplicaciones $\Phi : V(G) \rightarrow R$ y $\Phi' : E(G) \rightarrow R$. Si *G* es un digrafo, el grafo que se obtiene de *G* mediante la "eliminación de las flechas", se llama grafo base de *G*.

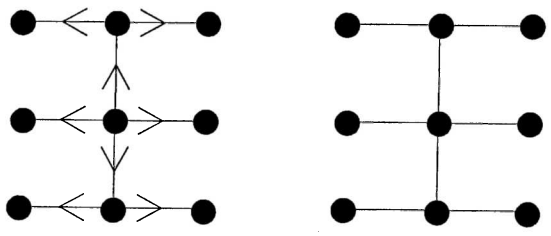

Figura 2.4.- Digrafo y grafo base del mismo

Las definiciones dadas para los grafos pueden extenderse de forma natural a los digrafos. Así, se dice que dos nodos *v* y *w* de un digrafo *G* son adyacentes si existe un arco de la forma *vw* o *wv* en *A(G)*. Se dice entonces que los nodos *v* y *w* son

incidentes en ese arco. Por otra parte, dos digrafos son isomorfos si existe un isomorfismo entre sus grafos base que preserve el orden de los nodos de cada arco. Asimismo una secuencia de arcos en un digrafo *D* es una secuencia finita de arcos de la forma v_0v_1, v_1v_2, ..., $v_{m-1}v_m$ en la que no se repiten ni arcos ni nodos.

Un digrafo *G* es conexo (o débilmente conexo) si el grafo base de *G* es un grafo conexo. Si, además de ello, para cada par de nodos *v* y *w* de *G* existe un trayecto de *v* a *w*, entonces se dice que *G* es fuertemente conexo. Si bien todo digrafo fuertemente conexo es conexo, no todos los digrafos conexos son fuertemente conexos. El digrafo de la Figura 2.4 es conexo pero no fuertemente conexo.

La diferencia entre digrafos conexos y fuertemente conexos resulta clara cuando se considera el plano de una ciudad cuyas calles tienen todas dirección única. Decir que el plano es conexo equivale a decir que podemos circular desde cualquier parte de la ciudad a otra, ignorando el sentido de circulación obligatorio de las calles. Por otro lado si el plano es fuertemente conexo, podremos circular desde cualquier parte de la ciudad a cualquier otra siguiendo siempre la dirección permitida.

2.2 *Árboles*

Se denomina bosque a un grafo que no posee ningún circuito o ciclo, y árbol a un bosque conexo. De este modo, un árbol es un grafo conexo sin ningún ciclo, como se muestra en la Figura 2.5.

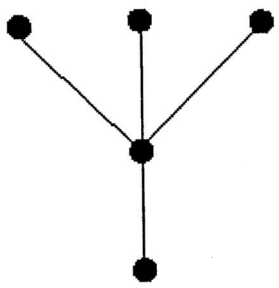

Figura 2.5.- Árbol

Un árbol tiene varias propiedades, de las cuales la más significativa es que dado un par de nodos existe un único camino que los une. Puede demostrarse ([12] y [22]) que las siguientes proposiciones, referidas a la definición de árbol y sus propiedades, son equivalentes:

(i). T es un árbol formado por *n* nodos.
(ii). T no contiene ningún ciclo, y posee *n-1* aristas.
(iii). T es conexo, y tiene *n-1* aristas.
(iv). T es conexo, y cada arista es un istmo, es decir, la eliminación de una arista divide el grafo en dos partes conexas. Estas partes no contienen además ciclos.
(v). Cada par de nodos de T está conectado por un único camino.
(vi). T no contiene ningún ciclo, pero la adición de cualquier nueva arista crea exactamente un ciclo.

Un árbol generador de un grafo *G(V,E)* se define como un subgrafo que contiene todos los nodos de *G* y que además es un árbol. De este modo, si el grafo *G* tiene *n* nodos, un árbol generador de *G* es siempre un subgrafo conexo de *n* nodos, *n-1* aristas y que además no contiene ciclos, es decir, en el cual existe un camino único para cada par de nodos de *G*.

2.3 *Árbol generador mínimo: algoritmo de Kruskal*

Como se ha dicho anteriormente, un grafo ponderado o etiquetado es un grafo *G(V,E)*, en el que cada arista tiene asignado un número llamado *etiqueta* o *peso* según una aplicación $\Phi : E(G) \rightarrow R$. El *peso* de un grafo etiquetado se calcula como la suma de los pesos de todas sus aristas.

Un árbol generador mínimo de un grafo conexo *G* con pesos es un árbol generador de *G* que tiene peso mínimo. El árbol generador mínimo de un grafo se suele denotar como MST, que es el acrónimo de la expresión inglesa *minimum spanning*

tree. Por definición, todo grafo conexo tiene un árbol generador mínimo, aunque en general no es único, como en el caso de la Figura 2.6.

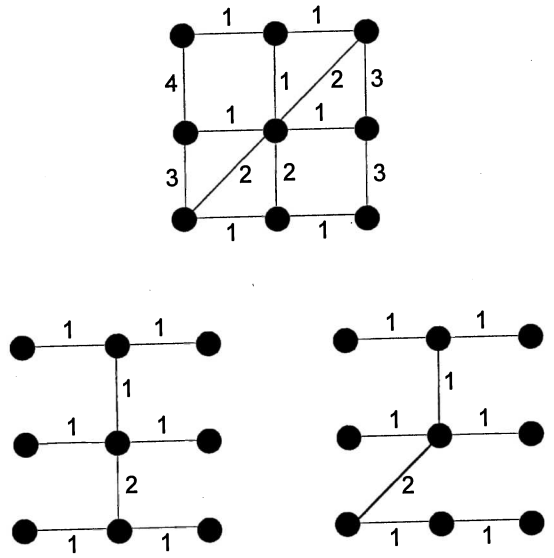

Figura 2.6.- Grafo y árboles generadores mínimos

De los algoritmos capaces de encontrar el árbol generador mínimo de un grafo conexo, el más empleado es el algoritmo de Kruskal, cuyo procedimiento es muy simple. Inicialmente, el algoritmo ordena las aristas por su peso. A continuación se van eligiendo las aristas por orden de peso –de menor a mayor– de tal forma que la arista escogida en cada iteración no puede formar ciclo con las aristas anteriores (de esta forma el conjunto de aristas finales no formará ciclos, y por lo tanto será un árbol). Para ello, cada vez que se escoge una arista, el algoritmo ha de actualizar en una tabla los conjuntos de nodos comunicados entre sí. De esta forma, el algoritmo no podrá seleccionar las aristas que unan nodos de un mismo subconjunto, porque de ese modo se estaría formando un ciclo. El proceso termina cuando se han escogido *n-1* aristas, en donde *n* es el número de nodos del grafo.

Por construcción el grafo resultante ha de ser un árbol generador mínimo: será un árbol porque sus aristas no forman ciclos; será generador porque esas *n-1* aristas han de unir necesariamente los *n* nodos del grafo (si unieran menos nodos, formarían obligatoriamente ciclos); y será mínimo porque por construcción no pueden construirse árboles de peso menor. Intuitivamente puede verse que para obtener una solución mejor deberíamos sustituir una arista *uv* de Kruskal por otra *u'v'* de menor peso, pero en ese caso la arista *u'v'* formaría ciclo con otras aristas seleccionadas anteriormente puesto que de otro modo, dicha arista *u'v'* no hubiera sido rechazada por Kruskal.

Supóngase, para ser más precisos, que T es el árbol generador de G calculado por Kruskal, y que T_1 es el árbol generador mínimo de G y supóngase además que de todos los árboles generadores mínimos, T_1 es el que tiene el mayor número de aristas comunes con T. Supóngase que T y T_1 son distintos, en cuyo caso Kruskal no habrá obtenido la solución óptima. Por otro lado, sea e la primera arista considerada por Kruskal que está en T pero no está en T_1. Sean C_1 y C_2 las componentes conexas de T que conecta la arista e. Puesto que T_1 es un árbol, $T_1 + e$ tiene un ciclo y existe una arista diferente f en ese ciclo que también conecta un nodo de C_1 con un nodo de C_2. Esta arista f no puede pertenecer a T, ya que en ese caso T contendría un ciclo.

Entonces $T_2 = T_1 + e - f$ es también un árbol generador. Ya que e fue considerada por Kruskal antes que f, el peso de e es menor o igual al peso de f. Y puesto que T_1 es un árbol generador mínimo, los pesos de esas dos aristas deben ser finalmente iguales (en otro caso, si el peso de e fuera menor al de f, entonces podría sustituirse e por f en T_1 y obtendríamos un árbol generador de peso menor, contradiciendo la hipótesis de que T_1 es mínimo). Por tanto, T_2 es un árbol generador mínimo con más aristas en común con T que las que tiene T_1, contradiciendo las hipótesis inicial para T_1. Esto prueba que T debe ser un árbol generador de peso mínimo. Igualmente, repitiendo el argumento, se podría obtener un nuevo árbol generador mínimo T_3 a partir de T_2, que tuviera con T una arista más en común que T_2, y así sucesivamente

hasta terminar construyendo *T* como árbol generador mínimo. En [12] puede encontrarse una demostración más formal del carácter óptimo del algoritmo.

```
T = Ø
 mientras |T| < n-1 hacer
      escoger e ∈ E de peso mínimo
      E = E - {e}
      si e no forma ciclo en T entonces
         T = T ∪ {e}
```

Figura 2.7.- Algoritmo Kruskal

Cabe añadir, por último, que la complejidad del algoritmo de Kruskal, tal y como se define en el siguiente capítulo, es de $O(m \cdot log\ m)$ en donde *m* es el número de aristas. Como el número de aristas es a lo sumo $n(n-1)/2$ si denotamos como *n* el número de nodos, puede afirmarse que la complejidad de Kruskal es a lo sumo de $O(n^2 \cdot log\ n^2) = O(n^2 \cdot 2 \cdot log\ n) = O(n^2 \cdot log\ n)$ que es además menor a $O(n^3)$. Es decir, la complejidad del algoritmo es polinómica.

Una forma simple de demostrar esto es contar en primer lugar el número máximo de iteraciones, que es igual al número *m* de aristas. Y en cada iteración, es decir, cada vez que se escoge o no una arista, sólo hay que comprobar si la arista tiene como nodos incidentes un par de nodos que ya han sido conectados por las aristas seleccionadas anteriormente y a continuación juntar en un mismo grupo conexo los dos grupos conexos a los que estaban conectados los dos nodos que forman la arista escogida. Puede demostrarse [11] que esta operación requiere un número de cálculos de orden $O(log\ m)$ y de ahí resulta la complejidad $O(m \cdot log\ m) = O(n^2 \cdot log\ n) < O(n^3)$.

3. Optimización combinatoria

En el presente capítulo se presentan los fundamentos de la optimización combinatoria. En primer término se define la familia de problemas NP-completos, a los cuales pertenece el problema planteado en el presente proyecto, y a continuación se presentan algunos mecanismos empleados en este tipo de problemas para encontrar soluciones aproximadas en tiempos de computación acotados.

3.1 *Problemas NP-completos*

Se dice que una función $G(n)$ es de orden superior a otra función $F(n)$, si existe algún entero n_0 tal que $\forall n > n_0$ se cumple que $G(n) > k \cdot F(n)$ para un número k positivo. En la tabla 3.1 se comparan valores de distintas expresiones en función de n. Puede observarse que las expresiones polinómicas crecen a un ritmo mucho menor que las funciones exponenciales o la función factorial. Se dice asimismo que una función $f(n)$ es $O(g(n))$, es decir, es del mismo orden que la función $g(n)$, si el cociente de las dos funciones $f(n)/g(n)$ puede acotarse para todo n arbitrariamente grande.

En la práctica se considera que un algoritmo es eficiente cuando es capaz de encontrar una solución óptima del problema en una función de tiempo polinómico con relación al tamaño del problema (en una red suele tomarse el número de nodos como tamaño). En ocasiones, esto no es posible. En estos casos se considera entonces que un algoritmo de resolución es aceptable si es capaz de dar una aproximación de la solución óptima en un intervalo de tiempo polinómico o bien si en la mayoría de casos es capaz de encontrar la solución óptima en un tiempo polinómico de ejecución.

Función	*n = 2*	*n = 8*	*n = 128*	*n = 1024*
n	2	2^3	2^7	2^{10}
$n\log_2 n$	2	$3 \cdot 2^3$	$7 \cdot 2^7$	$10 \cdot 2^{10}$
n^2	2^2	2^6	2^{14}	2^{20}
n^3	2^3	2^9	2^{21}	2^{30}
2^n	2^2	2^8	2^{128}	2^{1024}
$n!$	2	$4.9 \cdot 2^{13}$	$4.5 \cdot 2^{714}$	2^{8769}

Tabla 3.1.- Complejidad de distintas funciones

Un problema combinatorio cuya resolución requiere un algoritmo de complejidad *O(p)* (es decir un algoritmo que realice un número de operaciones proporcional a una función polinómica del tamaño del problema) se dice que es un problema de tipo P. De otro lado, un problema es de tipo NP cuando puede ser resuelto polinómicamente por una máquina de Turing no determinista, que puede definirse como un conjunto indefinido de máquinas de Turing procesando la información en paralelo. En este caso la complejidad temporal en función del tamaño *n* del problema viene determinada por el número máximo de operaciones que, considerando todas las posibles entradas de longitud *n*, debe realizar en el peor de los casos alguna de las máquinas de Turing. Una solución de un problema NP puede ser comprobada en un intervalo de tiempo polinómico.

Un problema de tipo P pertenece por definición a la clase NP. Se ha conjeturado de otro lado que P ≠ NP es decir, que existen problemas de tipo NP que no pueden ser resueltos en un tiempo de carácter polinómico.

Se dice que un problema combinatorio se puede transformar polinómicamente en otro problema si dada una solución de uno de los problemas puede construirse en un tiempo polinómico una solución del segundo problema. Se define de este modo la

clase de problemas NP-completos como un subconjunto de problemas de tipo NP tales que pueden ser traducidos unos en términos de los otros en un tiempo polinómico. Si pudiera resolverse eficientemente (es decir, mediante un algoritmo de complejidad polinómica) uno de los problemas de tipo NP-completo se demostraría que toda la familia puede ser resuelta en intervalos de tiempo polinómicos, dado que la composición de dos polinomios da como resultado otro polinomio.

Algunos de los problemas clásicos de tipo NP-completo pertenecen a la teoría de grafos, como el problema del conjunto independiente máximo, el aplanamiento de grafos, el coloreado de grafos o el problema del viajante de comercio. Estos problemas no pueden ser resueltos genéricamente con algoritmos de orden polinómico. O para ser más precisos, no se han encontrado por el momento técnicas polinómicas para su resolución. En estos casos deben definirse algoritmos capaces de encontrar soluciones casi óptimas en intervalos de ejecución polinómicos. Estos métodos, que suelen denominarse *métodos heurísticos* o de *optimización combinatoria*, aprovechan en cada caso las características del espacio de soluciones para acercarse eficientemente a las zonas en donde existen soluciones óptimas del problema. Algunos de estos algoritmos, como por ejemplo los algoritmos genéticos, simulated annealing o el algoritmo hormigas, responden además a esquemas generales y probabilistas (con el objeto de evitar extremos locales de la función de coste) que pueden ser aplicados sobre una amplia gama de problemas combinatorios.

3.2 *Optimización combinatoria*

La búsqueda exhaustiva del espacio de soluciones en un problema NP-completo es, como se ha dicho, un método computacionalmente inviable. Una simplificación, para ahorrar cómputos y tiempo, podría explorar únicamente una parte del espacio de soluciones, pero el método no sería eficiente y contendría además un alto carácter aleatorio. Lo que puede hacerse en estos casos es escoger la parte del espacio que se va a explorar de forma eficiente, buscando en los lugares en donde va a resultar

más probable encontrar buenas soluciones. Esto es lo que hacen los métodos heurísticos, también conocidos como algoritmos de optimización combinatoria.

Entre las estrategias utilizadas por los métodos heurísticos destacan las técnicas constructivas, los métodos de partición y los de mejora iterativa. Las técnicas constructivas estudian las características del espacio de soluciones y aprovechan este conocimiento para poder encontrar la solución óptima.

Por otra parte, los métodos de partición dividen el problema en un conjunto de subproblemas más pequeños, para los cuales es relativamente sencillo encontrar la solución. Más tarde generan, a partir de las soluciones encontradas para cada uno de los subproblemas, la solución global. Estos métodos son efectivos, pues, sólo si los subproblemas generados son disjuntos.

Finalmente los métodos de mejora iterativa son los más interesantes de todos ya que se pueden aplicar a distintos tipos de problemas. El más conocido de estos métodos es el de la búsqueda local. Este método parte de una solución cualquiera, escogida al azar, en la cual se fuerzan cambios sucesivos. Cada vez que se genera una nueva solución se almacena si es mejor que las anteriores y en caso contrario se rechaza.

Existen, por otro lado, los algoritmos de optimización combinatoria probabilistas de entre los cuales podemos destacar las redes neuronales, los algoritmos genéticos, el simulated annealing y el algoritmo hormigas. Estos algoritmos, que están basados en propiedades de sistemas reales, son probabilistas en el sentido de que algunas de sus operaciones se realizan bajo una determinada probabilidad. Es decir, no son algoritmos enteramente deterministas.

Esta naturaleza probabilista no implica en todo caso un desorden, ya que aunque los procesos realizados por estos algoritmos son estocásticos, la búsqueda que efectúan dentro del espacio de soluciones está dirigida. Así, no se deben confundir

con los algoritmos de búsqueda aleatoria pura. El uso de factores estocásticos tiene como objeto principal evitar los extremos locales en el espacio de soluciones.

Así, una de las principales características de estos métodos es la posibilidad de evitar los máximos o mínimos locales. Si el espacio de soluciones de un cierto problema tiene una característica como la de la Figura 3.1 sería muy fácil que un algoritmo de mejora sucesiva quedara estancado en alguno de los picos locales, mientras que con los algoritmos de optimización combinatoria probabilista la probabilidad de llegar al máximo global aumenta considerablemente, dado que permitiremos al algoritmo efectuar saltos de una zona a otra del espacio de soluciones.

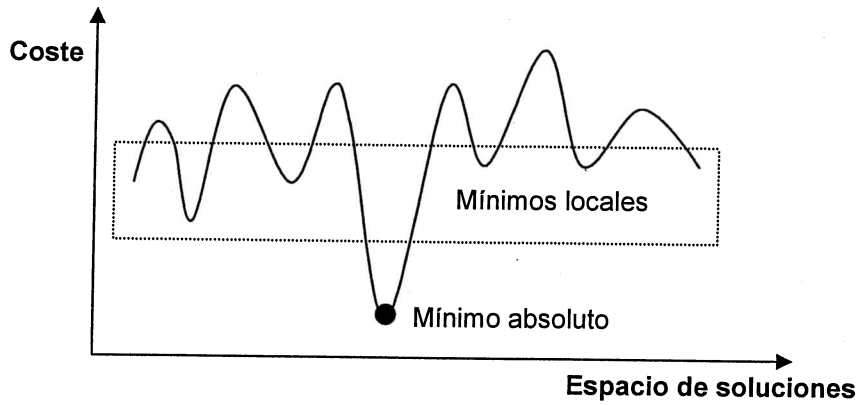

Figura 3.1.- Mínimos locales y absoluto de una función de coste

De esta forma, uno de los puntos clave para el buen funcionamiento de estos algoritmos es la representación del espacio de soluciones. La Figura 3.1 muestra un ejemplo sencillo, ya que normalmente el espacio resultante no puede ser ni tan siquiera dibujado. De todas formas, es interesante obtener un espacio de soluciones, sea cual sea su dimensión, sin extremos locales o, como mínimo, con extremos locales mucho peores que la solución óptima. Por ello, a veces resulta conveniente introducir un cierto escalado, ya que así los algoritmos utilizados convergerán más

rápidamente hacia la mejor solución y no perderán tiempo explorando los mínimos locales.

3.3 *Simulated annealing*

Si se reduce bruscamente la temperatura de un líquido por debajo del punto de fusión, el resultado es un estado desordenado de energía mayor que la que corresponde al compuesto en estado cristalino. En este caso, las moléculas han alcanzado un mínimo local de energía. Si, por el contrario, la temperatura del líquido se reduce lentamente y de acuerdo con una pauta adecuada de enfriamiento, el líquido evoluciona hacia el equilibrio y de esta forma su configuración, de energía mínima, es la de un compuesto ordenado y cristalino.

Inspirándose en este procedimiento, el algoritmo de simulated annealing trabaja con una única solución del problema que cambia en cada iteración de forma aleatoria (normalmente de forma restringida en el sentido de que modifica sólo una de las posiciones del vector de soluciones). Una vez obtenida la nueva solución, simulated annealing compara las energías (es decir las funciones de coste) de ambas soluciones. Si la energía de la nueva solución es menor que la energía de la antigua solución (supondremos un problema de minimización sin pérdida de generalidad) entonces la nueva solución es aceptada y sustituye a la anterior. Si, por el contrario, la energía ha aumentado la solución se acepta con una probabilidad determinada por el factor de Boltzman $e^{-\Delta f/T}$ en donde Δf es la diferencia de energías entre el nuevo estado j y el estado anterior i, y en donde T es la temperatura actual del sistema. De este modo:

$$P_{aceptar}(j) = \begin{cases} 1 & \text{si } f(j) \leq f(i) \\ e^{-\frac{f(j)-f(i)}{T}} & \text{si } f(j) > f(i) \end{cases}$$

La probabilidad de aceptar una nueva solución peor que la actual solución es, de este modo, menor cuanto mayor es la diferencia de energías, y menor cuanto menor es la temperatura del sistema.

Este proceso de generación y aceptación o refutación se repite un cierto número de veces para cada temperatura. Una vez se ha completado el número de iteraciones el sistema se enfría (es decir, se disminuye la temperatura del sistema), repitiéndose a continuación el mismo proceso hasta que se alcanza la temperatura final. Por otro lado, puede determinarse un número mínimo de aceptaciones para cada temperatura, de tal forma que si al llegar a la última iteración no se ha llevado a término un número mínimo de cambios el proceso continúa hasta alcanzar una cantidad mínima de cambios o bien hasta superar un número máximo de iteraciones. En este punto el algoritmo puede cambiar a una nueva temperatura o dar por finalizada la búsqueda.

Conforme el algoritmo se aproxima a la temperatura mínima la probabilidad de aceptar un aumento de la energía se aproxima a cero. Esta dependencia con respecto a la temperatura permite escapar de los mínimos locales y asegurar la convergencia del algoritmo. De hecho, se puede demostrar [2] que simulated annealing conduce al máximo global cuando el número de iteraciones crece indefinidamente.

Para el buen funcionamiento del algoritmo es importante ajustar los parámetros de funcionamiento, especialmente la temperatura inicial y el factor de enfriamiento. Una temperatura inicial elevada obliga al algoritmo a aceptar empeoramientos de la función de coste con una frecuencia alta de manera que la energía oscilará mientras la temperatura no haya superado un determinado umbral. De esta forma, aunque la solución encontrada finalmente pueda ser aproximadamente la misma, el número de iteraciones habrá aumentado con el consiguiente empeoramiento de la eficiencia del algoritmo. En cambio, la elección de una temperatura inicial demasiado baja comportará la no aceptación –o aceptación con una probabilidad casi nula– de

empeoramientos de la función de coste, aumentando la probabilidad de estancamiento en mínimos locales. La temperatura final debe ser tal que un determinado empeoramiento de la función de coste sea aceptado con una probabilidad comparable a cero. Esta temperatura debe asegurar además un número mínimo de estados explorados.

Inicio
inicializa temperatura $T=T_o$
busca una solución inicial al azar
calcula función de coste
para cada iteración
genera una nueva solución
calcula función de coste
si nueva solución mejor antigua solución
guarda nueva solución
si no
guarda nueva solución con probabilidad $e^{-\Delta f/T}$
si nueva solución menor que mejor solución
guarda nueva solución como mejor solución
si número de iteraciones igual a cambio de temperatura
cambia temperatura $T_k = rT_{k-1}$
fin para
hasta máximo número de iteraciones o solución encontrada

Figura 3.2.- Simulated annealing

Entre la primera y la última temperatura debe establecerse una pauta de enfriamiento. Si la probabilidad de aceptación de estados disminuye de forma exponencial en función de la temperatura y se desea que el cambio de probabilidad de aceptación al enfriarse el sistema sea significativo, la temperatura debe ser reducida según pautas exponenciales. De esta forma, el esquema de enfriamiento suele seguir la fórmula $T_k = T_0 \cdot r^k$, en donde T_k es el valor de la temperatura una vez

el sistema se ha enfriado k veces, T_0 es la temperatura inicial y r es la razón de enfriamiento, que acostumbra a tener valores próximos a la unidad.

Finalmente, el número de iteraciones que el algoritmo debe realizar para cada una de las temperaturas debe garantizar una exploración adecuada del espacio de soluciones, con el objeto de que el algoritmo no quede encerrado en un mínimo local.

3.4 *Algoritmo hormigas*

Aunque fue definido según otros criterios, el algoritmo hormigas puede entenderse como un algoritmo de mejora sucesiva –muy parecido al simulated annealing– en el que cada vez que se escoge una nueva solución no se hace de forma aleatoria sino conforme un criterio de optimización local. Con el objeto de no perder eficiencia este criterio de decisión local ha de ser simple. De otro lado, el algoritmo puede permitir eventualmente, lo mismo que simulated annealing, empeoramientos de la función de coste con el propósito de no caer en extremos locales.

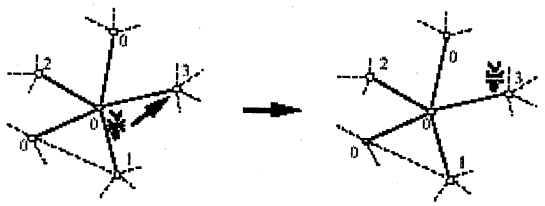

Figura 3.3.- Movimiento de una hormiga

A continuación se describe el funcionamiento del algoritmo en un problema de coloreado de grafos para el que fue originalmente descrito. En el capítulo siguiente, se describe con mayor detalle la adaptación del algoritmo para el contexto del presente proyecto: el problema del subgrafo de difusión de energía mínima.

En el problema del coloreado de grafos el algoritmo hormigas asigna inicialmente un color a cada nodo del grafo y distribuye asimismo una serie de agentes –llamados hormigas– a lo largo del grafo, todo de forma aleatoria. A continuación, cada una de las hormigas modifica el coloreado del grafo conforme un criterio de optimización local. De este modo, cada hormiga se desplaza en cada iteración hasta el nodo adyacente con una función de coste mayor –en este caso, al nodo cuyo color es utilizado por un número mayor de vecinos– y sustituye su color por un nuevo color con el objeto de minimizar la función de coste del nodo, es decir, el número de infracciones cometidas –que no siempre puede ser cero– en la asignación del nuevo color.

Inicio
colorea aleatoriamente el grafo
distribuye aleatoriamente las hormigas
para cada hormiga
si probabilidad p_n mueve al peor nodo adyacente
si no mueve a otro nodo adyacente
si probabilidad p_c asigna el mejor color
guarda nueva solución como mejor solución
si no asigna cualquier color
actualiza función de coste
si nueva solución menor que mejor solución
guarda nueva solución como mejor solución
fin para
hasta máximo número de iteraciones o solución encontrada

Figura 3.4.- Algoritmo hormigas para el coloreado de grafos

Este proceso se repite en cada iteración de forma aleatoria: cada agente u hormiga se desplaza al peor nodo adyacente con una determinada probabilidad p_n –en otro caso puede desplazarse a cualquier otro nodo adyacente– y le asigna el mejor color también bajo otra probabilidad p_c –en otro caso puede asignarle cualquier color

escogido de forma equiprobable. Ambas probabilidades son parámetros ajustables. El proceso de movimiento y coloreado de cada agente u hormiga se lleva a cabo de forma simultánea al movimiento de otras hormigas. El proceso se repite hasta que el algoritmo encuentra la solución óptima.

El número de hormigas que se desplaza a lo largo del grafo es un parámetro ajustable en función del diámetro del grafo y debe ser lo suficientemente grande como para que se produzca cooperación pero no excesivamente grande, a fin de asegurar que las hormigas no se interfieran continuamente. En este algoritmo, por tanto, es fundamental la idea de cooperación –prestada de la teoría de la complejidad– en el sentido de que las operaciones de las distintas hormigas se afectan –como sucedería en un hormiguero real– recíprocamente con el objeto de mejorar la eficiencia final del programa.

4. Subgrafo de difusión de energía mínima

Como se ha dicho en capítulos anteriores, el presente proyecto tiene como objeto la asignación de los radios de cobertura de los nodos de una red ad-hoc con el objeto de garantizar la conectividad de la red, de una parte, y de minimizar el consumo de energía, de otro. Dado que los enlaces son inalámbricos se ha empleado como función de coste el sumatorio de los radios de cobertura al cuadrado según la expresión $\sum R_i^2$. En este capítulo se describe el problema teórico correspondiente a dicha asignación y se describe el empleo de las técnicas presentadas en el capítulo anterior para su resolución.

4.1 *Definición*

Dado un cuadrado C de lado l, un conjunto S de n puntos o nodos distribuidos sobre C y un nodo raíz $s \in S$, el problema del subgrafo de difusión de energía mínima –MECBS según las siglas en inglés: Minimum Energy Consumption Broadcasting Subgraph– consiste en asignar a cada nodo i de S un radio de cobertura R_i de tal modo que pueda enviarse cualquier información desde el nodo raíz s a cualquier otro nodo de S y de que el sumatorio $\sum R_i^2$ sea mínimo. Dicho sumatorio refleja, como se ha dicho al principio de la memoria, el consumo total de energía, lo cual tiene especial importancia en las redes ad-hoc.

Con el fin de resolver el problema, se ha definido en cada caso un grafo dirigido en el cual un nodo u es adyacente a un nodo v si y sólo si el radio R_u de u es igual o mayor a la distancia que los separa, según se describe en la Figura 4.1. De este modo, una vez cada nodo tiene asignado un radio de cobertura, puede comprobarse si es posible mandar la información del nodo raíz al resto de nodos estudiando la conectividad del digrafo resultante.

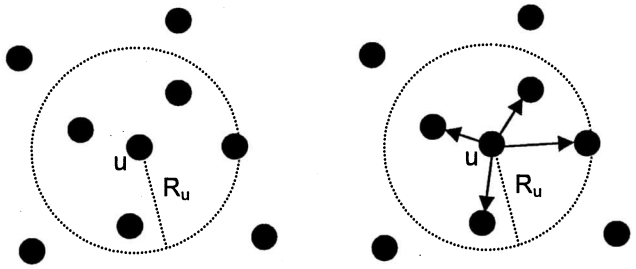

Figura 4.1.- Construcción del grafo a partir del radio de cobertura de los nodos

Se ha demostrado [6] que el problema del subgrafo de difusión de energía mínima pertenece a la clase de problemas NP-completos, lo que significa que no puede resolverse de forma óptima con algoritmos de complejidad polinómica. Es por eso que en los siguientes apartados se presentan algunas posibilidades para su resolución aproximada, que se evalúan comparadamente en el capítulo final de esta memoria.

4.2 *Algoritmo de Kruskal*

El problema del subgrafo de difusión de energía mínima puede resolverse de forma aproximada calculando el árbol generador mínimo del grafo, mediante el algoritmo de Kruskal, y difundiendo la información del nodo raíz al resto de la red a través de dicho árbol mínimo. En este caso, se ha considerado en primer lugar el grafo completo en el cual todos los nodos se pueden comunicar con todos los nodos a través de sus enlaces inalámbricos. Es decir, en un primer instante se ha supuesto que el radio de cobertura de todos los nodos es suficientemente grande como para cubrir al resto de la red.

Una vez definido el grafo completo, se ha calculado el árbol generador mínimo mediante el algoritmo de Kruskal, tal y como se ha descrito en el segundo capítulo. Hecho esto, se ha especificado la ruta de transmisión de datos siguiendo el camino

del nodo raíz al resto de nodos a través del árbol generador mínimo. Esta operación permite señalar para cada nodo de la red una serie de nodos a los que ha de retransmitir la información procedente del nodo raíz. A continuación se ha asignado a cada nodo un radio de cobertura igual a la distancia del nodo más lejano al que tiene que transmitir la información. De este modo, nos aseguramos de un lado conectividad, dado que por definición un árbol es un grafo conexo y por tanto podremos mandar la información desde el nodo raíz al resto de la red; y, por otro, también tendremos un consumo bajo de energía ya que emplearemos como referencia el árbol de peso mínimo de toda la red, es decir, el árbol con una función de coste menor si se suman todas las aristas del árbol final.

Cabe decir, con todo, que este mecanismo no permite obtener la solución óptima del problema del subgrafo de difusión de energía mínima, que es el problema que en verdad deseamos resolver. De hecho, se trata de problemas distintos. Mientras el problema del subgrafo de difusión de energía mínima MECBS es, como se ha dicho, un problema NP-completo, el problema del árbol generador mínimo MST puede resolverse con un algoritmo como Kruskal de complejidad polinómica. O dicho de otra forma, aunque Kruskal obtiene la solución óptima del MST, no puede garantizar la solución óptima para el MECBS, ya que se trata de problemas distintos, por mucho que su descripción sea muy parecida.

En la Figura 4.2 se expone un grafo sencillo para ilustrar la diferencia de los dos problemas. Se trata de una red cuadrada de lado igual a dos, formada por nueve nodos. Puede observarse que en este caso la solución obtenida con el árbol generador mínimo (MST) mediante el algoritmo de Kruskal –abajo a la izquierda– difiere notablemente de la solución del problema del subgrafo de difusión de energía mínima (MECBS), que en este caso se puede calcular a mano, dado el reducido tamaño de la red.

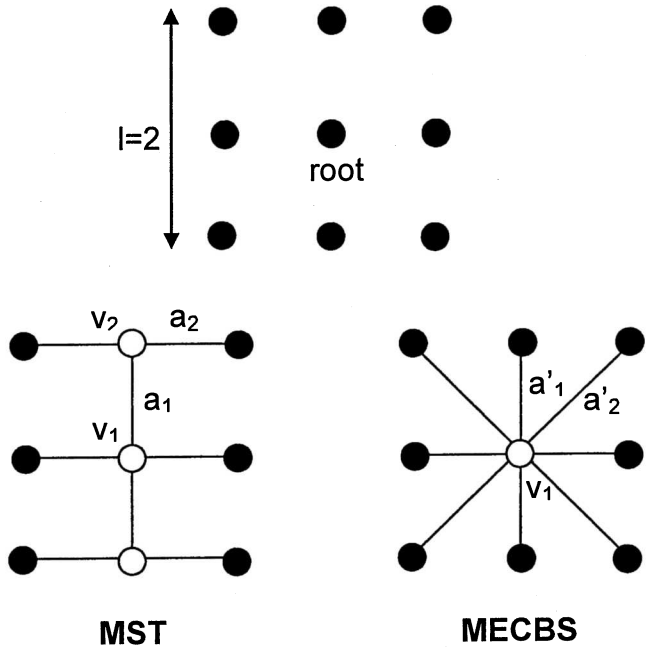

Figura 4.2.- MST y MECBS: sólo los nodos blancos han de mandar información

Como se observa en la figura, todas las aristas del MST tienen peso uno, dado que todas tienen longitud igual a la unidad, mientras que en el MECBS tenemos cuatro aristas de peso igual a uno (las dos horizontales y las dos verticales) y otras cuatro aristas de peso 2 (las aristas diagonales, puesto que su longitud es la raíz cuadrada de dos). Recuérdese que el peso de una arista es igual al cuadrado de la distancia de los dos nodos incidentes de la arista, es decir, la longitud de la arista al cuadrado. De este modo, es evidente que la suma de los pesos de todas las aristas es menor en MST, donde todas suman 8, que en MECBS, donde todas suman 12. Y ello, entre otros motivos, porque por definición el MST es el árbol de menor peso de todos los árboles generadores del grafo. Ahora bien, si por otro lado se calcula en los dos casos el sumatorio de los radios al cuadrado, en el caso del MST tendremos $\sum R_i^2=3$,

dado que los tres nodos dibujados en blanco tendrán que alcanzar otro nodo a través de un enlace de longitud uno, mientras que en el caso del MECBS resultará $\sum R_i^2=2$, dado que únicamente el nodo raíz o v_1 tendrá que cubrir a otros nodos mediante un enlace de longitud igual a la raíz de dos.

Dicho de otra manera, en MECBS podemos tener dos aristas a'_1 y a'_2 de mayor peso en conjunto (1 y 2 respectivamente) que otras dos aristas a_1 y a_2 (ambas de peso 1) en MST, a pesar de lo cual en MECBS podremos cubrir las dos aristas con el radio de un solo nodo v_1, mientras que en MST podríamos estar obligados a contar las dos aristas como radios de dos nodos distintos v_1 y v_2. La clave reside, por tanto, en que con un solo radio podremos cubrir en ocasiones más de una arista en el problema del MECBS. Y lo que deseamos en este caso es minimizar la suma de los radios, no la suma de las aristas. De esta forma, en el problema del MECBS podremos tener finalmente aristas con mayores pesos pero una suma de radios menor –dado que un solo radio podrá cubrir en algún caso más de una arista– y por tanto con menor consumo de energía. De ahí, como decíamos, que se trate de problemas distintos y que Kruskal sirva para obtener soluciones óptimas sólo en el caso del árbol generador mínimo, pero no en el del MECBS.

Es por esto que en el presente proyecto se ha aplicado el algoritmo de Kruskal únicamente como método de resolución aproximado del MECBS, y ello con el objeto de tomarlo como referencia comparativa con los modelos heurísticos del simulated annealing y algoritmo hormigas.

4.3 *Simulated annealing*

Como se ha dicho en el capítulo anterior, el algoritmo de simulated annealing se inspira en la analogía entre el estado de un sistema físico de partículas –una sustancia líquida, por ejemplo– y el espacio de soluciones de un problema de optimización combinatoria. Si la temperatura de las moléculas de una sustancia líquida se reduce repentinamente por debajo de su punto de fusión, el resultado es un estado semicristalino cuya energía es superior a la del estado cristalino puro. De

hecho, puede decirse que en un caso así las moléculas se han organizado alrededor de un punto local de energía mínima.

Por el contrario, si la temperatura del líquido se redujera lentamente –en un proceso denominado "annealing" en inglés– alcanzando el equilibrio antes de cada enfriamiento posterior, el líquido cambiaría al estado sólido cristalino puro, correspondiente a un punto global de energía mínima. En la analogía con un problema de optimización combinatoria, la distribución de las moléculas del líquido se correspondería con el espacio de soluciones del problema, mientras que la energía del líquido equivaldría a la función de coste del problema. La temperatura se definiría entonces como un parámetro de control relacionado con la probabilidad de aceptar eventuales empeoramientos de la función de coste, según se ha especificado en el capítulo anterior.

De este modo, dado el problema del subgrafo de difusión de energía mínima y un conjunto S de puntos dentro de un cuadrado C, el algoritmo de simulated annealing asigna en primer lugar un radio a cada nodo de tal manera que pueda mandarse la información desde el nodo raíz s al resto de la red S, es decir, de tal manera que el grafo resultante sea conexo. A continuación, el algoritmo escoge en cada iteración un nodo de forma aleatoria y cambia su radio de cobertura, también aleatoriamente. Si el nuevo radio no permite la difusión de la información a toda la red, entonces es rechazado y se conserva el radio anterior. Si por el contrario, el nuevo radio permite la difusión y mejora además la función de coste, es decir, es menor que el radio anterior (y por lo tanto requiere un consumo menor de energía), entonces es aceptado y sustituye al anterior. Si, por último, el nuevo radio permite la difusión de la información pero por otro lado empeora el consumo de energía porque es mayor que el anterior, entonces es aceptado bajo una probabilidad $e^{-\Delta f/T}$, en donde Δf es la diferencia entre la nueva y la antigua función de coste y en donde T es el parámetro de control al que llamamos *temperatura*. De esta manera, cuanto mayor sea el empeoramiento de la función de coste y menor la temperatura del sistema, menor será la probabilidad de aceptar una nueva solución peor a la anterior.

El algoritmo repite la misma operación un número determinado de veces para cada temperatura: cada vez se escoge aleatoriamente un nuevo nodo y le asigna un nuevo radio de cobertura, que es aceptado o rechazado según las condiciones descritas. Una vez concluida la última iteración para una temperatura dada, esta se reduce multiplicándola por un factor menor pero próximo a la unidad. De este modo, la temperatura va reduciéndose con el tiempo y disminuye por tanto la probabilidad de aceptar peores soluciones. El proceso se repite de forma que el sistema se enfría progresivamente hasta que converge.

4.4 *Algoritmo hormigas*

El algoritmo hormigas es un sistema multiagente en el cual distintas unidades llamadas *hormigas* trabajan en paralelo con el objetivo de resolver un determinado problema de optimización combinatoria. En el problema del subgrafo de difusión de energía mínima el algoritmo hormigas –del mismo modo que simulated annealing– asigna en un principio un radio de cobertura a cada nodo de forma que pueda mandarse la información del nodo raíz s al resto de la red. A continuación el algoritmo distribuye de forma aleatoria una serie de agentes u hormigas sobre la red.

En cada iteración, una hormiga se desplaza desde su posición actual hasta el peor nodo adyacente, es decir, hasta el nodo adyacente de mayor radio de cobertura. Dicho nodo se describe como "peor nodo" dado que es el que consume mayor cantidad de energía. A continuación, la hormiga asigna al nuevo nodo el menor radio posible de modo que el grafo resultante siga siendo conexo, es decir, de modo que siga siendo posible mandar la información desde el nodo raíz al resto de la red. Estas dos operaciones –desplazarse hasta el peor nodo adyacente y asignarle a continuación el menor radio posible– se realizan bajo unas determinadas probabilidades p_n y p_c con el objeto de evitar los mínimos locales. En caso de no asignar al nodo el menor rango posible, el algoritmo aumenta dicho rango en una unidad, con el objeto de admitir eventualmente pequeños empeoramientos y evitar de este modo los extremos locales. Por otro lado, en aquellos casos en los que la

hormiga no se desplaza al peor nodo adyacente, se sitúa sobre cualquier otro nodo escogido de forma aleatoria. Una vez la hormiga ha finalizado ambas operaciones, el proceso se repite para las restantes hormigas y así sucesivamente hasta que el algoritmo converge.

El algoritmo hormigas puede entenderse como una evolución del algoritmo de simulated annealing en el sentido de que en cada iteración los dos algoritmos repiten las mismas operaciones: escoger un nodo y asignarle un nuevo radio. La diferencia entre ambos programas es que simulated annealing realiza estas operaciones de forma completamente aleatoria, mientras que el algoritmo hormigas trata de aplicar a dichas decisiones algún conocimiento de la naturaleza del problema. Este conocimiento, beneficioso para resolver el problema, no puede ser sin embargo excesivamente voluminoso puesto que entorpecería la velocidad de convergencia del algoritmo. En nuestro caso, el algoritmo hormigas se informa en cada paso de cuál es el peor nodo adyacente y del menor radio que puede asignarle sin perder la conectividad del grafo. De este modo, las mejoras sucesivas se toman bajo un criterio simple de optimización, y no completamente a ciegas como sucede en simulated annealing.

4.5 *Ordenación de los nodos*

Tal y como ha sido definido al principio del presente capítulo, el problema del subgrafo de difusión de energía mínima tiene como objeto asignar radios de cobertura a los distintos nodos de una red ad-hoc con el propósito de cubrir toda la red, de un lado, y de minimizar el consumo de energía, de otro. La resolución del problema en un caso así puede no obstante simplificarse si se tiene en cuenta lo siguiente.

Supóngase que se ha asignado al nodo i un radio R_i de tal modo que dicho nodo cubre los nodos $i_0,i_1,...,i_r$, ordenados de modo creciente según su distancia al nodo i, es decir, de modo que $d(i,i_0) \leq d(i,i_1) \leq ... \leq d(i,i_r) \leq R_i$. Asúmase ahora que R_i es mayor que $d(i,i_r)$. En este caso, podríamos cubrir exactamente los mismos nodos asignando

a *i* un radio de cobertura igual a su distancia $d(i,i_r)$ a i_r, radio que es menor que R_i y que permitiría cubrir igualmente toda la red reduciendo además el consumo de energía, que es el objeto del presente proyecto. De este modo, finalmente se asignará a cada nodo *i* no un radio de cobertura sino un rango que indica el nodo i_r más alejado al que llega la señal de *i*. En ese caso, el radio correspondiente al nodo *i* será la distancia $d(i,i_r)$ entre *i* e i_r y el nodo *i* cubrirá los nodos $i_0,i_1,...,i_r$, es decir, el conjunto de nodos de la red cuya distancia a *i* es menor o igual a la distancia $d(i,i_r)$ que separa *i* de i_r.

Con el fin de simplificar los cálculos, se ha descrito al principio de hormigas y de simulated annealing, para cada nodo *i*, una lista con el resto de nodos ordenados conforme su distancia creciente a *i*. De este modo, en cada momento no se ha asignado a cada nodo *i* un radio R_i sino un nodo i_r que nos ha permitido obtener, sin necesidad de hacer cálculos, tanto el radio de cobertura de *i*, igual a $d(i,i_r)$, como el conjunto de nodos a los que cubre *i*, conjunto formado por la unión de i_r más los nodos que en la lista de *i* aparecen antes que i_r.

5. Resultados

Dado el amplio rango de problemas a los que pueden adaptarse, en el presente proyecto se han aplicado los algoritmos de hormigas y simulated annealing al problema del subgrafo de difusión de energía mínima. En este capítulo se presentan los resultados obtenidos con ambos métodos, que se comparan además con el modelo determinista planteado por Kruskal, según se ha descrito en el capítulo anterior.

5.1 *Pruebas*

Una vez programados el algoritmo hormigas y simulated annealing, se han realizado pruebas sobre un cuadrado de lado igual a la unidad, para redes que han variado desde los 10 hasta los 150 nodos. Dado que los nodos se han distribuido aleatoriamente y de forma uniforme a lo largo del cuadrado, los resultados habrían sido los mismos con cuadrados mayores, fuera del cambio de escala. Como referencia, se han comparado los resultados con los que se obtienen mediante un método simple que calcula el árbol generador mínimo del grafo mediante el algoritmo de Kruskal. En este punto conviene recordar que, aunque el algoritmo Kruskal resuelve el problema del árbol generador mínimo de forma óptima, ofrece peores resultados que los algoritmos de hormigas y simulated annealing, puesto que son problemas distintos. Es decir, Kruskal obtiene de forma simple la solución óptima del problema del árbol generador mínimo, pero eso no garantiza la solución óptima del problema del subgrafo de difusión de energía mínima ya que, como se ha descrito en el capítulo anterior, se trata de problemas cuya descripción es muy parecida pero cuya solución puede diferir considerablemente.

Para un determinado número de nodos n se han creado cinco redes aleatorias, distribuyendo los n puntos de forma equiprobable a lo ancho de un cuadrado de lado uno. Sobre estas redes se han aplicado los algoritmos de Kruskal, simulated

annealing y hormigas. En el caso de simulated annealing se ofrecen dos columnas, *sa9* y *sa15*, según las iteraciones empleadas en cada caso para una temperatura determinada: en la primera columna se han ejecutado $9 \cdot 10^4$ iteraciones para cada temperatura, mientras que en la segunda se llegaba a las $15 \cdot 10^4$ iteraciones. Es por esto que en la columna *sa15* los resultados son algo mejores, si bien el algoritmo ha tardado más en converger. En ambos casos la temperatura inicial ha tomado un valor de $3 \cdot 10^{-2}$, la temperatura final de 10^{-4} y el factor de decrecimiento ha sido igual a 0.99. Es decir, después de completar las iteraciones para una temperatura, esta se ha multiplicado cada vez por un factor de 0.99. Esto supone un total de 568 enfriamientos dado que $10^{-4} \approx 3 \cdot 10^{-2} \cdot 0.99^{568}$. De este modo, el número total de iteraciones ha sido de $568 \cdot 9 \cdot 10^4 = 51'12 \cdot 10^6$ en el primer caso y de $568 \cdot 15 \cdot 10^4 = 85'20 \cdot 10^6$ en el segundo. El algoritmo se ha detenido al alcanzar la temperatura final.

La selección de las temperaturas inicial y final se he realizado conforme criterios estadísticos, teniendo en cuenta que la probabilidad de escoger una solución peor que la anterior se calcula como $e^{-\Delta f/T}$. En nuestro caso, como la temperatura inicial es de $3 \cdot 10^{-2}$, la probabilidad de que el radio al cuadrado de un nodo empeore en un valor de 0.01 se ha calculado como $e^{-0.01/0.03} = 0.71$, lo cual tiene sentido ya que, como se ha dicho, al principio se permiten ciertos empeoramientos con el objeto de evitar los mínimos locales. Para esta temperatura inicial la probabilidad de empeorar la función de coste en un 0.02 sería de $e^{-0.02/0.03} = 0.51$ y en un 0.04 de $e^{-0.04/0.03} = 0.26$. Por otro lado, al final de todo, la probabilidad de aceptar soluciones peores ha de ser casi nula. En nuestro caso la probabilidad de aceptar un empeoramiento de valor 0.01 es, si la temperatura final se acerca a 10^{-4}, de $e^{-0.01/0.0001} = 3'72 \cdot 10^{-44}$, que es una cantidad despreciable en comparación con el número total de iteraciones para dicha temperatura, que como se ha dicho vale $9 \cdot 10^4$ o $15 \cdot 10^4$.

En el caso del algoritmo hormigas se han empleado, después de distintas tentativas, tres hormigas y unas probabilidades p_n y p_c de valor 0.85. El algoritmo se ha

detenido después de $12 \cdot 10^6$ iteraciones, una cifra comparable con las $85'20 \cdot 10^6$ iteraciones totales de simulated annealing, si se tiene en cuenta que en hormigas se realizan más operaciones durante cada iteración.

5.2 *Resultados*

En la Tabla 5.1 se muestran los resultados obtenidos para cada uno de los métodos aplicados. Como se ha dicho, para simulated annealing se han considerado dos posibilidades según el número de iteraciones para cada temperatura fuera de $9 \cdot 10^4$ o $15 \cdot 10^4$. Para poder interpretar con mayor facilidad el comportamiento de los distintos algoritmos, en la Figura 5.1 se ofrece la representación gráfica de los resultados de la Tabla 5.1.

nodos	Kruskal	hormigas	sa9	sa15
10	0.481141	0.408792	0.406025	0.406009
20	0.596848	0.424671	0.406636	0.406577
30	0.536641	0.420764	0.406479	0.406455
40	0.494716	0.418308	0.391767	0.392513
50	0.527738	0.414592	0.375313	0.373196
60	0.522188	0.420086	0.381584	0.379628
70	0.503625	0.429504	0.385939	0.378004
80	0.517248	0.442937	0.391605	0.390174
90	0.507378	0.466318	0.379718	0.374101
100	0.521337	0.442226	0.384641	0.382632
110	0.485787	0.454347	0.374818	0.373626
120	0.489317	0.453126	0.375937	0.372230
130	0.492154	0.469436	0.380448	0.378129
140	0.461308	0.446637	0.360974	0.360094
150	0.474975	0.478231	0.375884	0.374686

Tabla 5.1.- Soluciones en función del número de nodos

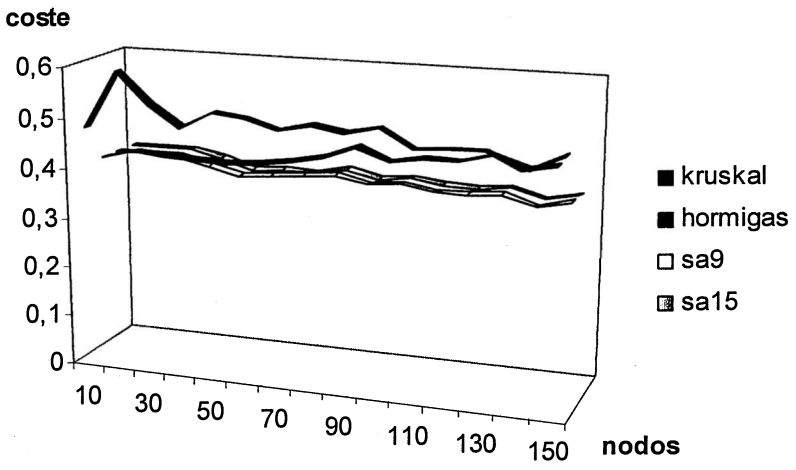

Figura 5.1.- Soluciones en función del número de nodos

Como se ha dicho, dado un valor *n* del número de nodos se han realizado para cada algoritmo cinco pruebas sobre cinco redes distintas. En la tabla se ofrece el resultado promediado de la solución obtenida en los veinticinco casos: cinco soluciones para cinco redes (Kruskal, en todo caso, se ha aplicado una sola vez ya que como es determinista siempre ofrece los mismos resultados). Recuérdese que dicha solución se ha calculado como la suma del radio de los cuadrados de todos los nodos, es decir $\sum R_i^2$.

En primer lugar, se observa que tanto simulated annealing como el algoritmo hormigas ofrecen mejores resultados que el algoritmo determinista de Kruskal, con diferencias que han llegado al 20%. Por otro lado, los resultados de la columna *sa15* son ligeramente mejores que los de la columna *sa9* dado que en el primer caso el número de iteraciones por temperatura ha sido ligeramente mayor, y por lo tanto se ha producido una búsqueda más exhaustiva del espacio de soluciones. Como precio, sin embargo, el algoritmo tarda más en converger en *sa15*.

Por otra parte, puede comprobarse que para un número reducido de nodos, el algoritmo hormigas ofrece prácticamente los mismos resultados que simulated annealing, pero conforme aumenta el tamaño de la red se incrementan las diferencias, siempre a favor de simulated annealing. Esto es así dado que el coste computacional de simulated annealing no depende del número de nodos –la búsqueda de nuevas soluciones se hace siempre a ciegas– mientras que el algoritmo hormigas tendrá que realizar mayores operaciones conforme aumenta el tamaño de la red. Recuérdese que, como se ha explicado en el capítulo anterior, el algoritmo hormigas realiza los cambios puntuales basándose en la evaluación del entorno de cada nodo, y ese entorno aumentará con el número de nodos de la red. Es por esto que al crecer el tamaño de la redes hormigas se ha ido alejando de los resultados de simulated annealing.

Por otro lado, cabe señalar que la solución final se mantiene casi constante para los tres algoritmos, en función del número de nodos. Esto es así puesto que aunque al aumentar el número de nodos tendremos que mandar información a un mayor número de destinatarios, por otro lado habrá aumentado la densidad de la red y por lo tanto los nodos que retransmitan la información podrán emplear radios de cobertura menores, puesto que probablemente tendrán nodos a distancia menor. Como se observa en la gráfica, el aumento de densidad de la red compensa finalmente el aumento del número de destinatarios, de tal forma que la solución final permanece casi constante, salvo en el caso del algoritmo hormigas por las razones que se han explicado en el párrafo anterior. Durante la realización del proyecto se realizaron pruebas sobre círculos de diámetro igual a uno, obteniéndose resultados prácticamente idénticos, aunque con números algo menores dado que la superficie de un círculo de diámetro uno es un 21.5% menor que la superficie de un cuadrado de lado uno.

Por otro lado, para comparar el comportamiento de simulated annealing y de hormigas, la Figura 5.2 muestra la estadística del rango final de los nodos para las redes de 50 nodos. Recuérdese que el rango de un nodo se define como el número

de nodos a los que alcanza finalmente con su radio de cobertura. Los resultados mostrados en la gráfica se han vuelto a promediar para veinticinco casos: cinco soluciones distintas obtenidas sobre cinco redes de cincuenta nodos. Como puede verse, el comportamiento es prácticamente idéntico para los dos algoritmos. La mayoría de nodos, aproximadamente unos 30, tienen rango 0, es decir, no han de mandar información a ningún otro nodo. De otro lado, unos 9 nodos tienen un rango igual a 1, unos 5 tienen rango 2 y a partir de ahí el número de nodos empieza a decrecer exponencialmente conforme aumenta el rango. De hecho, a partir de un rango igual a diez casi no aparecen nodos. De modo que tendremos muchos nodos con rango pequeño y pocos nodos de rango grande, como era de esperar, en una distribución más o menos uniforme del esfuerzo de los distintos nodos de la red.

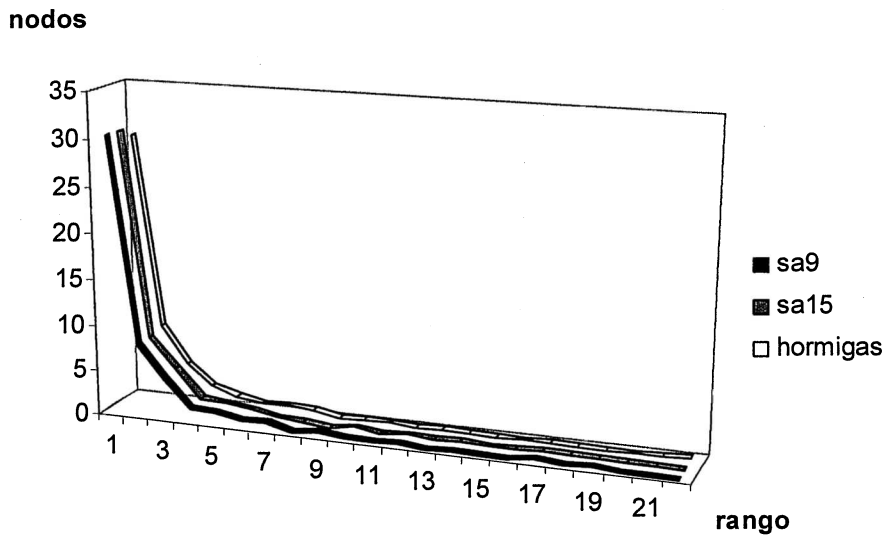

Figura 5.2.- Estadística del rango para redes de 50 nodos

En último lugar, para comprobar si la distribución paritaria de rangos en el caso de simulated annealing y hormigas, se traducía en una parecida distribución de los radios de cobertura asignados a los nodos, se han calculado las gráficas de la

Figura 5.3. En ella se han ordenado de forma creciente los 1250 radios al cuadrado que se han asignado en nuestras estadísticas: cinco soluciones para cinco redes en cada una de las cuales se tenía que asignar el radio a cincuenta nodos distintos, en total 5·5·50 = 1250 radios asignados. En este caso, con todo, más que el estudio de la distribución de los radios, interesa la comparación de las curvas para los dos algoritmos que, como se puede observar, vuelven a ser prácticamente idénticas. De la lectura de la gráfica puede observarse además que la práctica totalidad de los nodos van a tener un radio al cuadrado menor que 0.05, lo cual significa que la inmensa mayoría de los nodos no tendrán que alcanzar distancias significativas y por tanto no deberán consumir grandes cantidades de energía. Dicho de otro modo, la distribución de los radios seguirá una cierta uniformidad: tendremos muchos nodos de radios pequeños y pocos nodos de radios grandes, como era de esperar puesto que el objetivo final del proyecto ha sido disminuir el consumo de energía, es decir, el radio de cobertura de cada uno de los nodos de la red.

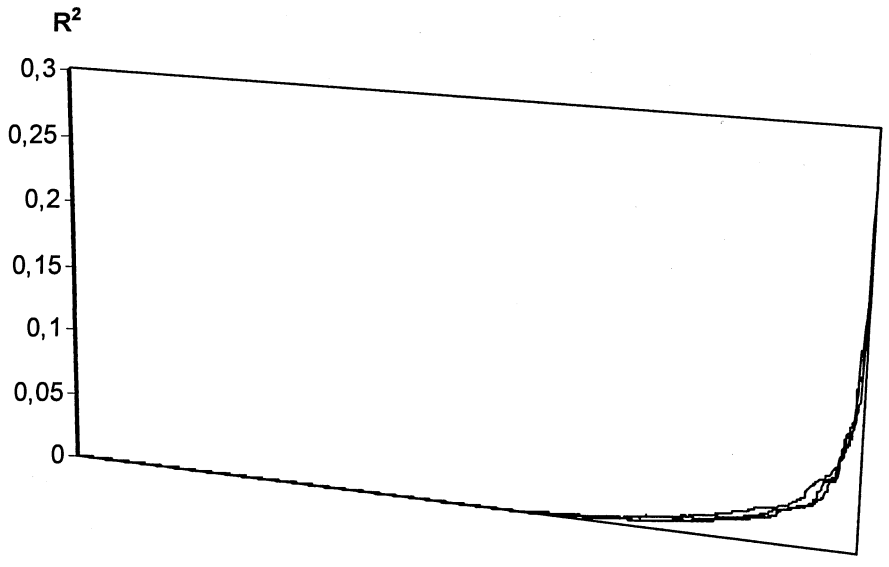

Figura 5.3.- Variación del radio al cuadrado para redes de 50 nodos

5.3 *Líneas futuras*

Dado que el presente proyecto ha constituido una primera tentativa para probar la conveniencia de los algoritmos heurísticos en el diseño de la topología de una red ad-hoc, existen numerosas líneas abiertas de investigación.

En primer lugar, se podría tratar de mejorar los actuales algoritmos sin modificar la definición del problema, cambiando algunas de sus características o buscando con más detalle los mejores parámetros de funcionamiento, especialmente en el caso del algoritmo hormigas, para el que se podrían además buscar las soluciones con mayores tiempos de computación: como se ha dicho anteriormente el algoritmo hormigas es más sensible que simulated annealing al tamaño de la red, y por esto para las redes grandes podría valer la pena darle más tiempo al algoritmo hormigas, con la esperanza de obtener mejores soluciones.

También podrían aplicarse nuevos algoritmos heurísticos de comparación, empezando por los algoritmos genéticos o las redes neuronales. En el caso de las redes pequeñas, se podrían comparar las soluciones planteadas con las soluciones óptimas reales, que podrían obtenerse mediante una exploración exhaustiva del espacio de soluciones. En este caso habría que comprobar el máximo número de nodos para el que un algoritmo de búsqueda exhaustiva puede determinar la solución en un tiempo de cálculo razonable.

Igualmente, podrían modificarse algunas de las características del propio problema con el fin de representar con mayor exactitud la verdadera naturaleza de las redes ad-hoc. En primer término, podría tratar de minimizarse, en vez del sumatorio $\sum R_i^2$ otro sumatorio $\sum R_i^D$ en donde el exponente $D \geq 2$ reflejase con mayor fidelidad el consumo real de energía de los nodos, que es proporcional a R_i^2 en el vacío pero que en medios físicos reales supera siempre dicho valor. En este caso, podría considerarse incluso la posibilidad de medir el verdadero exponente D de consumo sobre el terreno y emplearlo en los cálculos de la función de coste $\sum R_i^D$ de los algoritmos.

En el presente proyecto hemos considerado que un nodo debe enviar información al resto de la red. En un caso general, no obstante, podría estudiarse la posibilidad de que todos los nodos intercambiasen información entre sí, lo cual desembocaría en una topología ligeramente distinta y de mayor coste energético. Podría plantearse, además, la posibilidad de que los nodos no ocuparan una posición fija sino que se desplazaran a lo largo del espacio, en cuyo caso tendría que contabilizarse no sólo el consumo de energía de las distintas soluciones sino también la velocidad de convergencia de los algoritmos planteados. Esto es así dado que en un escenario variable habría que ir cambiando la topología de la red sobre la marcha, conforme los nodos fueran desplazándose a lo largo del espacio, y por tanto necesitaríamos algoritmos que plantearan soluciones óptimas o casi óptimas en intervalos reducidos de tiempo.

Por otra parte, dado que el consumo de energía de un nodo se incrementa de forma considerable con el radio de cobertura, el programa podría tener en cuenta la capacidad y el estado de las baterías de cada uno de los nodos durante la asignación de los radios de cobertura. De esta forma, los nodos con menor capacidad o con un tiempo de vida muy limitado por culpa del agotamiento de las baterías, quedarían asignados con radios de cobertura menores y verían aumentado su ciclo de vida. El resultado final sería que la red, en su conjunto, también incrementaría su ciclo de vida sin necesidad de recargar las baterías continuamente. Por último, en un plazo intermedio, se puede plantear la posibilidad de realizar medidas sobre redes reales.

Bibliografía

[1] E. Aarts, J.K. Lenstra (ed.), *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chicester, 1997.

[2] E. Aarts, J. Korst, *Simulated Annealing and Boltzman Machines*, John Wiley & Sons, Chicester, 1989.

[3] M. Abellanas, D. Lodares, *Análisis de Algoritmos y Teoría de Grafos*, RA-MA Editorial, Madrid, 1990.

[4] J. Abril, F. Comellas, A. Cortés, J. Ozón, M.Vaquer, *A multi-agent system for frequency assignment in cellular radio networks*, IEEE Trans. Vehic. Tech., vol. 49 (No. 5), pp.1558-1565, 2000.

[5] J.E. Baker, *Adaptative Selection Methods for Genetic Algorithms*, Proceedings of the First International Conference on Genetic Algorithms and their Applications, pp. 101-111, Lawrence Erlbaum Associates Publishers, 1985.

[6] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, P. Vocca, *On the Complexity of Computing Minimum Energy Consumption Broadcast Subgraphs*, Proceedings of STACS'01, 18th Ann. Symposium on Theoretical Aspects of Computer Science, LNCS, 2001.

[7] A. Colorni, M. Dorigo, V. Maniezzo, *Distributed optimization by ant colonies*, Proceedings of the First European Conference on Artificial Life Paris, pp. 134-142, ed. By F. J. Varela and P. Bourgine, MIT-Press-Bradford Books, Massachusetts, 1991.

[8] S. Colorni, M. Dorigo, V. Maniezzo, *An investigation of some properties of an ant algorithm*, Proceedings of the Second Conference on Parallel Problem Solving from Nature, pp. 509-520, Brussels, ed. By R. Maenner and B. Manderick, Amsterdam, 1992.

[9] D. Costa, A. Hertz, *Ants can colour graphs*, Journal of the Operational Research Society, vol. 48, pp. 295-305, 1997.

[10] M.R. Garey, D.S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H Freeman and Company, New York, 1979.

[11] A.M. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, Cambridge, 1985.

[12] J. Gimbert, R. Moreno, J.M. Ribó, M. Valls, *Apropament a la teoria de grafs i als seus algorismes*, Edicions de la Universitat de Lleida, Lleida, 1998.

[13] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishers, Reading, 1989.

[14] C. Gómez, J. Paradells, *Redes ad-hoc: el próximo reto*, Revista Burán, vol. 21, pp. 30-37, mayo 2004.

[15] D.S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, Boston, 1997.

[16] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, *Optimization by Simulated Annealing: an Experimental Evaluation*; Part II, Graph colouring and Number Partitioning, Operations Research, vol. 39, pp. 378-406, 1991.

[17] K.A. De Jong, W.M. Spears, *Using Genetic Algorithms to Solve NP-Complete Problems*, Proceedings of the Third International Conference on GA, pp. 124-132, Morgan Kauffman Publishers Inc., 1989.

[18] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, *Optimization by Simulated Annealing*, Science, vol. 220, pp. 671-680, 1983.

[19] J. Ozón, *Contribución al coloreado de grafos y las redes pequeño mundo*, Tesis Doctoral, Universitat Politècnica de Catalunya, 2001.

[20] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Boston, 2001.

[21] C.-K. Toh, *Ad Hoc Mobile Wireless Networks: protocols and systems*, Prentice-Hall, Upper Saddle River, NJ, 2002

[22] R.J. Wilson, *Introducción a la Teoría de Grafos*, Alianza Universidad, Alianza Editorial, Madrid, 1972.