Escola Politècnica Superior
de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Diseño de la topología de una red ad-hoc

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemas de telecomunicació

AUTOR: Maria Angel Lema Rosas

DIRECTOR: Javier Ozón Górriz

DATA: 23 de juliol de 2008

Título: Diseño de la topología de una red ad-hoc

Autor: María Angel Lema Rosas

Director: Javier Ozón Górriz

Fecha: 23 de julio de 2008

Resum

Las redes ad-hoc de comunicación están formadas por diferentes dispositivos, normalmente inalámbricos, que pueden situarse en cualquier punto del espacio. Para poder comunicarse se han de definir enlaces entre los diferentes nodos de manera que exista conectividad entre todos los nodos de la red.

Este proyecto, continuación de trabajos anteriores, tiene como objetivo el diseño de la topología de una red ad-hoc con un doble requerimiento. Por un lado, se ha de garantizar conectividad entre todos los nodos de la red (y ello bajo la posibilidad de utilizar cualquier dispositivo como nodo intermedio de conmutación). Por otro lado, se ha de conseguir que el consumo total de las baterías de los dispositivos sea mínimo.

El problema presentado por los anteriores proyectos, conocido como subgrafo de difusión de energía mínima (MECBS), fue resuelto mediante la utilización de algoritmos probabilistas –simulated annealing y hormigas– y un algoritmo determinista basado en la obtención de árboles generadores mínimos mediante Kruskal. En ese caso, un único nodo raíz debe mandar información al resto de la red. El presente proyecto trata de añadir pequeñas variaciones a los algoritmos de simulated annealing y hormigas para mejorar sus resultados y generaliza el problema para el caso en que todos los nodos deban enviar información al resto de nodos (problema que hemos denominado subgrafo de gossiping de energía mínima, MECGS), lo que ha de dar como resultado –en comparación con el caso en que sólo un nodo manda información al resto de la red– topologías ligeramente distintas y de mayor densidad de enlaces.

Title: Design of the topology of an ad-hoc network

Authors: María Angel Lema Rosas

Director: Javier Ozón Górriz

Date: July, 23th 2008

Overview

Ad-hoc communication networks are usually formed by wireless devices which can be located everywhere on the network. It is necessary to define the links between the different devices in real time in order to communicate each other with complete connectivity.

This project, based on previous investigations, has the aim of designing the topology of an ad-hoc network with a double requirement. On the one hand, we must ensure the total connectivity of the network (assuming the communication through devices which act as passing nodes). On the other hand, we must assure that the battery consumption of the devices is as little as possible.

The problem presented in previous projects, better known as Minimum Energy Consumption Broadcasting Subgraph (MECBS), was first solved by using two probabilistic algorithms –simulated annealing and ants algorithm– and a deterministic method based on Kruskal which first obtains the minimum spanning tree. In the case of MECBS, a single root node has to send information to the rest of the network. The aim of this project is to improve the probabilistic algorithms in order to reach better results. It will be also studied the case were all devices are in communication at a time (i.e. where all nodes sends information to the rest, called Minimum Energy Consumption Gossiping Subgraph, MECGS), which must obtain –comparing with the case of MECBS where only one node sends information– different topologies of ad-hoc networks with higher density of links.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1. REDES AD-HOC 3

1.1. Breve historia 3

1.2. Características 3

1.3. Protocolos en redes ad-hoc 5

1.4. Encaminamiento en redes ad-hoc 5

1.5. Aplicaciones 5

CAPÍTULO 2. TEORÍA DE GRAFOS 7

2.1. Definiciones 7

2.2. Árboles 10

2.3. Árbol generador mínimo: algoritmo de Kruskal 10

CAPÍTULO 3. OPTIMITZACIÓN COMBINATORIA 13

3.1. Problemas NP-completos 13

3.2. Optimitzación combinatoria 14

3.3. Simulated annealing 15

Algoritmo de Hormigas 17

CAPÍTULO 4. SUBGRAFO DE DIFUSIÓN DE ENERGÍA MÍNIMA 21

4.1. Definición 21

4.2. Algoritmo de Kruskal para MECBS 22

4.3. Algoritmo de Kruskal para MECGS 24

4.4. Ordenación de los nodos 26

4.5. Simulated annealing 26

4.6. Algoritmo de Hormigas 27

CAPÍTULO 5. RESULTADOS 29

5.1 Pruebas 29

5.2 Resultados MECBS 31
5.2.1 Resultados para simulated annealing 31
5.2.2 Resultados para hormigas..... 32
5.2.3 Comparativa 34

5.3 Resultados para MECGS 38

5.4 Comparativa MECGS y MECBS..... 39

5.5 Conclusiones 40

5.6 Líneas futuras 41

BIBLIOGRAFIA..... 43

APENDICE. SUBGRAFO DE GOSSIPING DE ENERGÍA MÍNIMA (MECGS) 45

1. Definición de MECGS..... 45

2. Resolución de MECGS mediante MST 46

3. MECGS ≠ MST 52

4. COTA DE MECGS 55

5. CONCLUSIONES..... 60

INTRODUCCIÓN

Una red de comunicaciones ad-hoc está formada por diferentes dispositivos móviles que pueden situarse en cualquier punto del espacio. Para poder comunicarse y asegurar la conectividad de la red es necesario crear enlaces inalámbricos entre los dispositivos. Por otro lado, el tiempo de vida de los terminales depende principalmente del estado de las baterías, de modo que el consumo de potencia es de vital importancia en las redes ad-hoc. De esta forma, si en un primer modelo de diseño se asignaran radios de cobertura a los distintos nodos de manera que todos cubrieran directamente al resto de puntos, se aseguraría de forma sencilla la conexión completa de la red, pero a cambio se tendría un enorme consumo de potencia, que disminuiría el tiempo de vida de los dispositivos. Igualmente, la transmisión de señales de alta potencia incrementaría las interferencias entre nodos, lo que se traduciría en una menor capacidad de la red. Es por esto que en las redes ad-hoc se intenta reducir en la medida de lo posible el radio de cobertura 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 comunicarse con los nodos más alejados la información deberá retransmitirse 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 (o todos, dependiendo del contexto, como veremos más adelante) pueda mandar información a los demás nodos de la red, no necesariamente de forma directa (i.e. de tal manera que la información pueda llegar a través de una serie de nodos intermedios, en funciones de conmutación). Y, en segundo lugar, lograr que el consumo total de energía sea mínimo. Por otra parte, y puesto que tenemos redes inalámbricas, consideraremos que el consumo de energía es proporcional al cuadrado de los radios de cobertura, de modo que finalmente trataremos de minimizar la suma de los cuadrados de los radios de cobertura de todos los nodos de la red.

En nuestro caso se han planteado dos escenarios fundamentales. En el primero, analizado en anteriores proyectos [4,20] y conocido como MECBS –*Minimum Energy Consumption Broadcast Subgraph*–, un nodo raíz ha de mandar información al resto de nodos de la red. Se ha demostrado [7] que este problema pertenece a la familia de problemas NP-completos, de forma que para su resolución se han aplicado métodos aproximados de optimización combinatoria (que en el presente proyecto, además, se han modificado para mejorar los resultados de los trabajos anteriores). En segundo lugar, se ha analizado el caso en que todos los nodos deben mandar información al resto de la red, problema que hemos denominado *Minimum Energy Consumption Gossiping Subgraph* (MECGS). En el apéndice se demuestra que este segundo problema tampoco puede resolverse mediante técnicas deterministas simples (basadas, como se verá, en la obtención de árboles mediante el algoritmo de Kruskal), de modo que se han aplicado los mismos métodos probabilistas empleados en el primer problema de MECBS. Por otro lado, en ambos casos se ha comparado el comportamiento de los algoritmos probabilistas con un método determinista simple basado en el algoritmo de Kruskal. Igualmente se han comparado, para las mismas redes, el consumo de energía requerido por MECBS (en el que un único nodo ha de mandar información al resto) con el consumo para MECGS (en el que todos los nodos mandan información al resto de la red). En este segundo caso, aunque el consumo es mayor, se ha visto que el coste total no es proporcional al número de nodos. Es decir, proporcionalmente resulta más rentable una red en la que todos los nodos intercambian información, que una red en la que se tiene un único nodo emisor.

La memoria se divide de la siguiente forma. En el primer capítulo se describen las principales características y aplicaciones de las redes ad-hoc. En el segundo capítulo se ofrece una breve introducción a la teoría de grafos (marco teórico necesario para el estudio analítico de las redes de comunicación). A continuación, en el tercer capítulo, se presentan las herramientas aproximadas para la resolución de los problemas y, en el cuarto, se especifica con más detalle los procedimientos aplicados en el contexto particular de este proyecto. Finalmente, en el quinto y último capítulo, se exponen los resultados obtenidos, se presentan las conclusiones y de describen las principales líneas futuras de investigación.

CAPÍTULO 1. REDES AD-HOC

El éxito de las comunicaciones inalámbricas y la progresiva reducción de la medida de los dispositivos para comunicar datos, ha incrementado el estudio de las redes ad-hoc formada por terminales móviles. En este capítulo se presentan las características generales de este tipo de redes, las cuales constituyen el marco de trabajo del presente proyecto.

1.1. Breve historia

El término ad-hoc proviene del latín y significa literalmente “para esto”, es decir, “con un propósito determinado”. Según esta definición, una red ad-hoc se constituye con un cierto propósito gracias a un conjunto de dispositivos independientes que pueden establecer entre ellos enlaces inalámbricos y conformar una red en ausencia de una infraestructura previa. La figura 1.1 muestra un ejemplo de red ad-hoc.

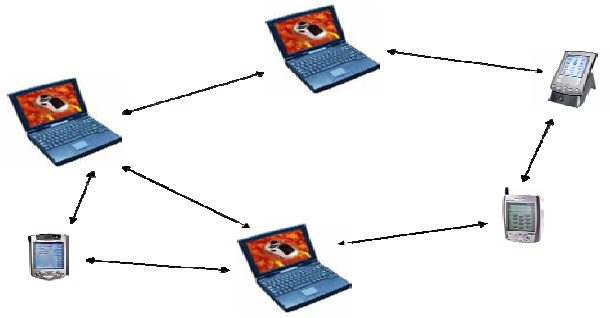


Fig. 1.1 Red ad-hoc

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), con el objetivo de 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. 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 sus numerosas aplicaciones en la sociedad civil.

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. Por este motivo, las redes ad-hoc tienen una

topología variable, de modo que cualquier nodo que forme un enlace con otro nodo puede moverse y salirse de su radio de cobertura y entrar a su vez en el radio de un nuevo nodo. La variación de la posición de los nodos obliga de este modo a un cambio constante de las rutas de encaminamiento, tal como se muestra en la figura 1.2.

Por otro lado, la naturaleza inalámbrica de los enlaces de una red ad-hoc se traduce normalmente en un menor ancho de banda que los enlaces fijos y en una mayor propensión a los errores de transmisión. Igualmente, hay que considerar que los dispositivos de una red ad-hoc pueden tener una autonomía limitada, debido a su reducida capacidad de batería. Esto obliga a transmitir a baja potencia, lo que se traduce en un bajo radio de alcance, que puede oscilar entre unos pocos 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. De este modo, en las redes ad-hoc no se distingue entre dispositivos terminales y nodos intermedios, de forma que cualquier nodo puede desempeñar ambos papeles, careciendo la red de una estructura jerárquica o centralizada.

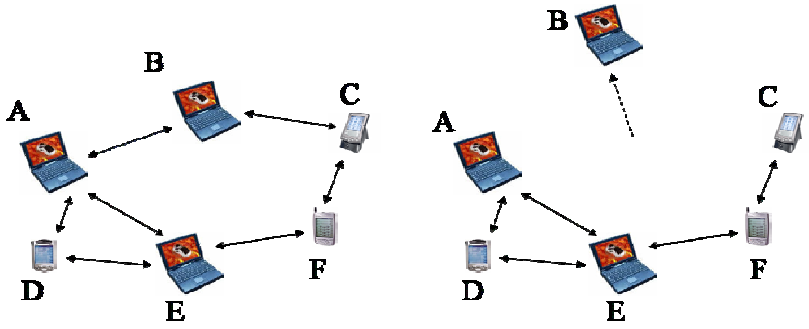


Fig. 1.2 El nodo B se desplaza y desaparece del área de cobertura de los nodos A y C: el tráfico de la ruta ABC deberá utilizar otra ruta, como AEFC

Debido a la duración limitada de las baterías de los nodos portátiles, los programas de control de la red han de poder 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 con el objeto de aumentar su ciclo de vida.

1.3. Protocolos en redes ad-hoc

En los últimos años la arquitectura TCP/IP ha mostrado una enorme flexibilidad para adaptarse a una gran variedad de contextos tecnológicos, propiedad que se ha traducido en la universalidad de la red basada en ambos protocolos: Internet. En rigor, Internet puede definirse como un conjunto de redes que pueden tener diferentes plataformas tecnológicas pero que son capaces de intercambiar información mediante el conjunto de protocolos TCP/IP. Es por esto que las redes ad-hoc pueden definirse también sobre la base de protocolos TCP/IP, hecho que además permite su sencilla integración en Internet. A pesar de esto, en las redes ad-hoc el protocolo TCP no siempre funciona correctamente por culpa de la falta de fiabilidad de las conexiones inalámbricas. TCP fue concebido para las comunicaciones por cable, que provocan menos errores de transmisión y suelen proporcionar más ancho de banda, además de permanecer inamovibles a lo largo del tiempo. Es por esto 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 las funciones de cliente y servidor, y puesto que dichos nodos pueden aparecer y desaparecer de la red debido a 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 la autentificación y privacidad, para prevenir ataques 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 además puede desplazarse de un punto a otro del espacio deshaciendo rutas existentes y 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 gran necesidad de cálculo, conforme los nodos se desplazan de un punto a otro de la red.

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

1.5. Aplicaciones

Dependiendo de la naturaleza de sus nodos pueden definirse dos tipos 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 diferentes 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 triangulación, propiedad que puede resultar de gran utilidad en situaciones de emergencia, como 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 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 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 especial importancia en las redes de sensores, en las que distintos nodos pueden transmitirse instantáneamente sus medidas. En un caso así, la información puede mostrarse finalmente de forma centralizada y en tiempo real, si todas las medidas de 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 unir uno o más edificios cercanos que forman parte normalmente de la misma empresa o campus universitario. En este escenario conviven nodos fijos alimentados por la red eléctrica y nodos móviles de tamaño reducido que se alimentan con una batería. Cualquiera de estos dispositivos puede actuar como puente entre otros nodos, pero los primeros deberían ser usados con mayor prioridad dado que no presentan el problema del agotamiento de las baterías. Por otro lado la propia infraestructura de la red puede proporcionar en estos casos 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 activara alguna alarma. E incluso podrían tomarse medidas estadísticas para optimizar el consumo de energía, tanto en verano como en inverno, conectado el aire acondicionado o la calefacción según unas pautas óptimas de ahorro energético.

CAPÍTULO 2. TEORÍA DE GRAFOS

En el siguiente capítulo se definen las herramientas teóricas necesarias para el problema de asignación de radios de cobertura en una red ad-hoc. Las redes se han representado mediante grafos, por lo que en este capítulo se definen los conceptos fundamentales de la teoría de grafos. Además, también se presenta el problema del árbol generador mínimo, que más adelante se ha empleado como método de resolución aproximada en el diseño de la topología de una red ad-hoc.

2.1. Definiciones

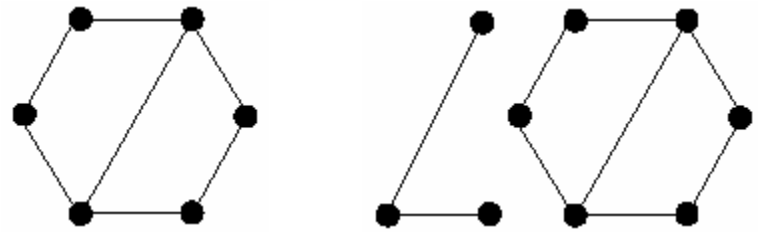
Un grafo simple *G* es una pareja *(V(G),E*(G)) tal que *V(G)* es un conjunto finito de elementos llamados nodos o vértices y *E(G)* un conjunto finito de parejas no ordenadas de nodos. Cada uno de los elementos de *E(G)* se llama arista. Tanto los nodos como las aristas de un grafo pueden incluir etiquetas definidas a partir de aplicaciones Φ:$V(G)\rightarrow R$ y Φ':$E(G)\rightarrow R$ 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)*. Igualmente, 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 puede representarse gráficamente dibujando los nodos como puntos y las aristas como líneas que unen los nodos, tal y como se observa en la figura 2.1 y siguientes.

Por definición, un grafo simple no puede tener aristas repetidas, es decir, parejas de nodos unidos por más de una arista, ni tampoco lazos (aristas que unen un mismo nodo). A partir de ahora, y a no ser que se especifique lo contrario, cada vez que nos refiramos a un grafo *G* entenderemos que es un grafo simple. Dos nodos *u* y *v* son adyacentes o vecinos cuando los une una arista *uv*. En este caso, se dice que los nodos *u* y *v* inciden sobre una arista *uv* o también 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 al nodo *v*, es decir, el número de nodos a los que está unido directamente por una arista. Para 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 posibles combinaciones que se pueden formar con los *n* nodos tomados de dos en dos. Así, se define 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 esta manera:

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

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 describe o dibuja un camino continuo sobre el grafo. Aquella secuencia en la que no se repiten aristas se define como cola y si tampoco se repite ningún nodo, trayecto o camino. Un trayecto cerrado tal que el primer y el último nodo coinciden se llama circuito o ciclo. Se dice que un grafo 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 se puede descomprimir un grafo no conexo se llama 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 se ha de recorrer para llegar a un nodo o a otro. Cuando un grafo no es conexo y dos nodos pertenecen a componentes diferentes del grafo se dice que la distancia entre ambos nodos es infinita. Se llama diámetro de un grafo *G* a la máxima distancia entre dos nodos cualquiera de *G*.



Grafo conexo Grafo inconexo

Fig 2.1 Ejemplo de grafo conexo y grafo inconexo

Un grafo completo K_n es un grafo simple tal que todas sus parejas de nodos están unidos por una arista. De esta manera, el número de aristas de K_n es $n(n-1)/2$. Por otro lado, un grafo en el que todos los nodos tienen el mismo grado se le llama grafo regular. Concretamente se dice que *G* es un grafo regular de grado *r* si todos sus nodos tienen grado *r*. Dado 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 el caso que G_2 conserve las aristas de G_1, es decir, en el caso que toda pareja de nodos de G_2 sean siempre adyacentes si lo son en G_1, decimos 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 toda pareja 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* a *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* a *G*. Un grafo $G=(V,E)$ es vértice-transitivo 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.

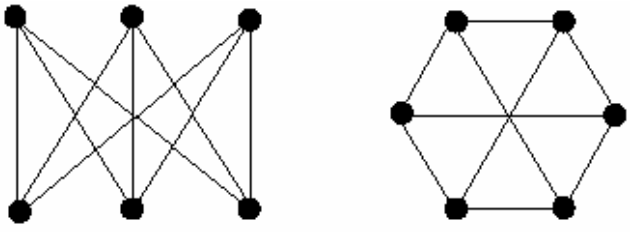


Fig 2.2 Grafos isomorfos vértice-transitivos.

Un grafo $G=(V,E)$ es plano si se puede dibujar sobre el plano sin que ninguna de sus aristas se corte. Un grafo $G=(V,E)$ es planar si existe algún grafo plano isomorfo a G. Un grafo *G* es k-partito o k-coloreable si existe una aplicación $\Phi:V(G)\rightarrow \{1,...,k\}$ tal que $\Phi(u)\neq\Phi(v)$ para toda pareja de nodos *u* y *v* adyacentes, es decir, tal que dos nodos

adyacentes tengan siempre imágenes diferentes. La aplicación Φ se llama coloreado de los nodos del grafo *G* y cada uno de los enteros asignados a cada nodo se llama color. El número cromático *X(G)* de un grafo *G* es el mínimo entero *k* tal que *G* es k-coloreable. De esta manera, el coloreado de un grafo consiste en asignar un color (o etiqueta o número) a cada uno de los nodos del grafo de tal forma que dos nodos adyacentes tengan colores diferentes. Igualmente, el número cromático de un grafo es el mínimo número de colores necesarios para colorearlo correctamente.

Un digrafo *G* se define como una pareja *(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 parejas ordenadas de elementos de *V(G)*, llamados arcos o aristas dirigidas o también, por simplicidad aristas. En las ocasiones en que no exista ambigüedad, se utiliza el término grafo para referirse 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 *vw*. Los arcos *wv* y *vw* son diferentes. Si *G* no posee ningún lazo (arcos de 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 mediante la "eliminación de las flechas" se llama grafo base de *G*.

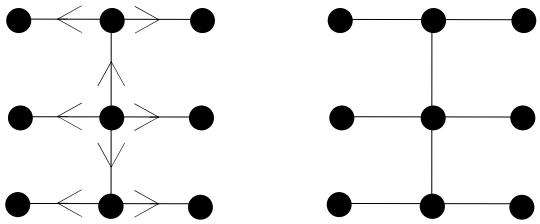


Fig. 2.3 Digrafo y base del mismo.

Las definiciones dadas para grafos pueden extenderse de forma natural a los digrafos. Así, se dice que dos nodos *v* y *w* 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 aquel arco. Una secuencia de arcos de un digrafo *G* es una secuencia finita de arcos de la forma v_0v_1, v_1v_2, v_2v_3, …, $v_{m-1}v_m$ en la que no se repiten ni arcos ni nodos. Un digrafo *G* es conexo (o débilmente conexo) si el grafo base *G* es un grafo conexo. Si, además para cada pareja de nodos *v* y *w* de *G* existe un trayecto de *v* a *w*, entonces se dice que *G* es fuertemente conexo. Todos los digrafos fuertemente conexos son conexos, pero no todos los conexos son fuertemente conexos. El digrafo de la figura 2.3 es conexo, pero no fuertemente conexo.

La diferencia entre digrafos conexos y fuertemente conexos resulta clara cuando se considera en un plano de una ciudad, en el cual todas las calles tienen dirección única. Afirmar que un plano es conexo equivale a decir que podemos circular desde un punto de la ciudad a cualquier otro, ignorando el sentido de la circulación obligatoria de las calles. En cambio si decimos que el plano es fuertemente conexo, podemos seguir circulando desde cualquier punto de la ciudad a cualquier otro, pero siempre respetando la dirección permitida.

2.2. Árboles

Se le llama bosque a aquel grafo que no posee ningún circuito o ciclo y, árbol, a cualquier bosque conexo. De ésta manera un árbol es un grafo conexo sin ningún ciclo, tal y como se puede observar en la figura 2.4.

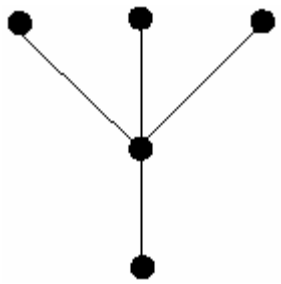


Fig. 2.4 Árbol

Un árbol tiene diversas propiedades, la más significativa de las cuales es que dada una pareja de nodos sólo existe un camino que los una. Se puede demostrar (véase [24]) 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 as un istmo, es decir, la eliminación de una arista cualquiera divide el grafo en dos partes conexas, cada una de las cuales no contiene ciclos.
v. Cada pareja de nodos de *T* está conectado por un único camino.
vi. *T* no contiene ningún ciclo, pero al añadir una arista se formará exactamente un ciclo.

Un árbol generador de un grafo *G=(V,E)* es un subgrafo que contiene todos los nodos de *G* y que además es un árbol. De esta manera, 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, que contiene un único camino para cada pareja de nodos.

2.3. Árbol generador mínimo: algoritmo de Kruskal

Como se ha dicho anteriormente, un grafo *ponderado* o etiquetado es un grafo *G,* 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 2.5.

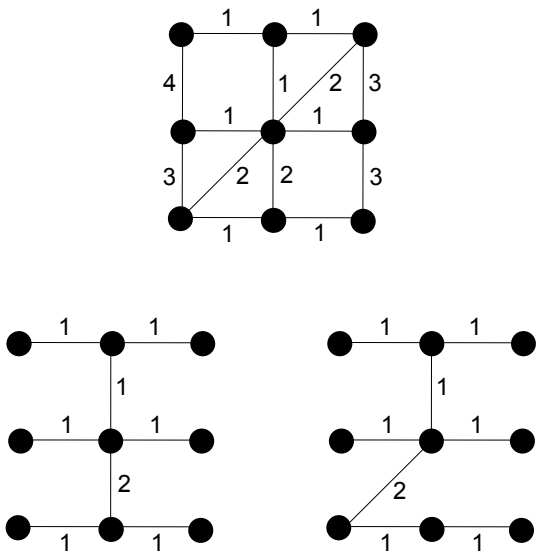


Fig. 2.5 Grafo y árboles generadores mínimos.

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 la iteración no puede formar un ciclo con las aristas anteriores (de esta forma, el conjunto de aristas finales no formará ciclos y por 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 dos nodos de un mismo subconjunto, porque de este modo estaría formando un ciclo. El proceso termina cuando se han escogido *n-1* aristas, en donde *n* es el número total de nodos del grafo.

Por construcción el grafo resultante ha de ser un árbol generador mínimo: es un árbol porque sus aristas no forman ciclos; es generador porque esas *n-1* aristas han de unir necesariamente los *n* nodos del grafo (si unieran menos nodos, formarían obligatoriamente ciclos); y es mínimo porque por construcción no pueden formarse árboles de peso menor. Intuitivamente puede verse que para obtener una solución mejor deberíamos sustituir la arista *uv* de Kruskal por otra *u'v'* de menor peso, pero en este 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 un árbol generador mínimo de G y supóngase además que de todos los árboles generadores mínimos, T_1 es el que tiene mayor número de aristas comunes con T. Supóngase que T y T_1 tienen distintos pesos, 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 este caso T contendría un ciclo.

Entonces $T_2=T_1+e-f$ es también 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 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}
```

Fig 2.6 Algoritmo de 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 del grafo. 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 aristas seleccionadas anteriormente y, a continuación, juntar en un mismo conjunto conexo los dos conjuntos conexos a los que estaban conectados los dos nodos que forman la arista escogida. Puede demostrarse [12] que esta operación requiere un número de cálculos de orden $O(\log m)$ y de ahí resulta la complejidad $O(m \cdot \log m) = O(n^2 \cdot \log n) < O(n^3)$.

CAPÍTULO 3. OPTIMITZACIÓN COMBINATORIA

En el presente capítulo se presentan los fundamentos de la optimización combinatoria. En primer término se define la familia de problemas NP-completos, al cual pertenece el problema del MECBS planteado en este 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.

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 las distintas funciones.

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 o de aristas 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.

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 P. Por 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 de máquinas 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 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 $P \neq NP$, 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 obtenerse 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 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 el producto 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 hasta 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 de 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. Optimització combinatoria

La búsqueda exhaustiva del espacio de soluciones de 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 este método no es eficiente y contiene 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. Finalmente los métodos de mejora iterativa son las 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 entre los cuales podemos destacar las redes neuronales, los algoritmos genéticos, el simulated annealing y el algoritmo de 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 deben confundirse 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 posibilidad de llegar al máximo local aumenta considerablemente, dado que permitiremos al algoritmo dar saltos de una zona a la otra del espacio de soluciones.

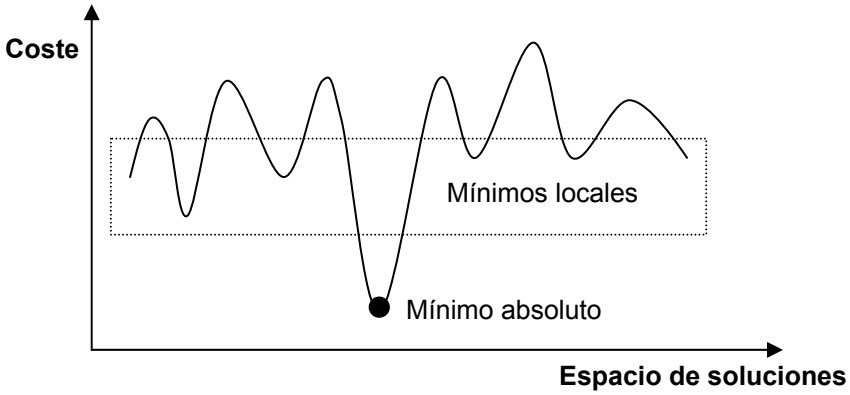


Fig 3.1 Mínimos locales y absoluto de la función de coste

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 y no perderán eficiencia explorando el espacio de soluciones próximo a un mínimo local.

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 de forma aleatoria cada iteración (normalmente de forma restringida en el sentido de que modifica solo 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 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 $f(i+1)-f(i)$ entre el nuevo estado $i+1$ y el estado anterior i, y en donde T es la temperatura actual del sistema. De este modo:

$$P_{aceptar}\,(i+1) = \begin{cases} 1 & \text{si } f(i+1) \le f(i) \\ e^{-\Delta f/T} = e^{-(f(i+1)-f(i))/T} & \text{si } f(i+1) > f(i) \end{cases}$$

La probabilidad de aceptar una nueva solución peor que la actual 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 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 [1] 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 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 la temperatura T=T0
- Busca una solución inicial al azar
- Calcula función de coste

Para cada temperatura

Para cada iteración

- Genera una nueva solución
- Calcula la función de coste

Si (nueva solución mejor**)**
- Guarda nueva solución

Si no
- Guarda nueva solución con probabilidad $e^{-\Delta f/T}$

Si (nueva solución menor que mejor solución**)**
- Guarda nueva solución

Si (número de iteraciones máximo**)**
- Cambia temperatura $\rightarrow T_k = r \cdot T_{k-1}$

Final de cada iteración

Hasta temperatura final ($T=T_f$) o solución encontrada

Fig 3.2 Simulated annealing

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

Algoritmo de Hormigas

Aunque fue definido según otros criterios, el algoritmo de hormigas puede entenderse como un algoritmo de mejora sucesiva –muy parecido al simulated annealing– en el que las nuevas soluciones no se escogen de manera aleatoria sino conforme a un criterio de optimización local. Con el objeto de no perder eficiencia, este criterio 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.

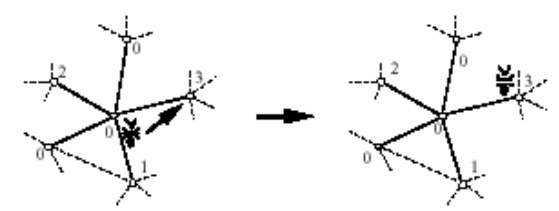


Fig 3.3 Movimiento de una hormiga

A continuación se describe el funcionamiento del algoritmo en el 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: subgrafos de energía mínima. En el problema del coloreado de grafos el algoritmo de 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 mayor número de vecinos– y sustituye su color por uno nuevo 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 de un nuevo color.

Inicio

- Colorea aleatoriamente un grafo
- Distribuye aleatoriamente las hormigas

Para cada hormiga

Si (Probabilidad P_n**)**
- mueve a peor nodo adyacente

Si no
- mueve a otro nodo adyacente

Si (Probabilidad P_c**)**
- asigna el mejor color

Si no
- Asigna cualquier color

Actualiza función de coste

Si (nueva solución menor que mejor solución**)**
- Guarda la nueva solución como mejor solución

Final para cada hormiga

Hasta número máximo de iteraciones o solución encontrada.

Fig 3.4 Algoritmo de 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 una 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.

CAPÍTULO 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, sobre dos escenarios distintos. En el primero, un nodo llamado raíz ha de enviar información al resto de la red. En el segundo, todos los nodos deben intercambiar información con el resto de nodos. 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 ΣR_i^2. En este capítulo se describen los dos problemas teóricos correspondientes a la asignación de radios en los dos escenarios definidos 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 desde el nodo raíz *s* pueda alcanzarse cualquier otro nodo de *S* y que el sumatorio ΣR_i^2 sea mínimo.

Por otro lado, en el caso en que todos los nodos de la red tengan que mandarse información entre sí, el problema se llamará MECGS –también según sus siglas en inglés: *Minimun Energy Consumption Gossiping Subgraph*. Formalmente, el problema del MECGS se define de la misma forma que el MECBS –se pretende minimizar el sumatorio ΣR_i^2– con la diferencia de que ahora se ha de asignar a los nodos un radio R_i tal que desde cualquier nodo pueda alcanzarse el resto de la red.

Para resolver ambos problemas se ha definido en cada caso un grafo dirigido en el que un nodo *u* es adyacente a un nodo *v* cuando el radio de cobertura R_u de *u* es mayor o igual a la distancia que separa ambos nodos, tal como se muestra en la figura 4.1. De este modo, una vez que se ha asignado a cada nodo un radio de cobertura puede comprobarse si es posible mandar la información del nodo raíz al resto de nodos (o para el caso del MECGS, desde todos los nodos al resto de la red) estudiando las conectividades del digrafo resultante.

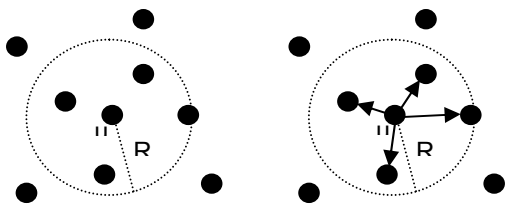


Fig 4.1 Construcción del grafo a partir del radio de cobertura de los nodos

Se ha demostrado [7] que el problema del subgrafo de difusión de energía mínima MECBS 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.

En el caso del MECGS, aunque existe la posibilidad de obtener la solución óptima mediante técnicas deterministas, más adelante se verá que en general los métodos simples basados en Kruskal no van a permitir tampoco obtener soluciones óptimas, por lo que también será necesario recurrir a algoritmos de optimización combinatoria.

4.2. Algoritmo de Kruskal para MECBS

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 (según se ha visto en el anterior capítulo), 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. Acto seguido, se ha establecido la ruta de transmisión de los datos siguiendo el camino del nodo raíz al resto de la red 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 un árbol de peso mínimo.

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.

En la figura 4.2 se expone un grafo sencillo para ilustrar la diferencia entre ambos problemas. Se trata de una red cuadrada de lado igual a dos, formada por 9 nodos. Se puede observar que en este caso la solución obtenida con el árbol generador mínimo difiere de la solución del problema del subgrafo de difusión de energía mínima MECBS, que en este caso se puede calcular a mano ya que se trata de una red pequeña.

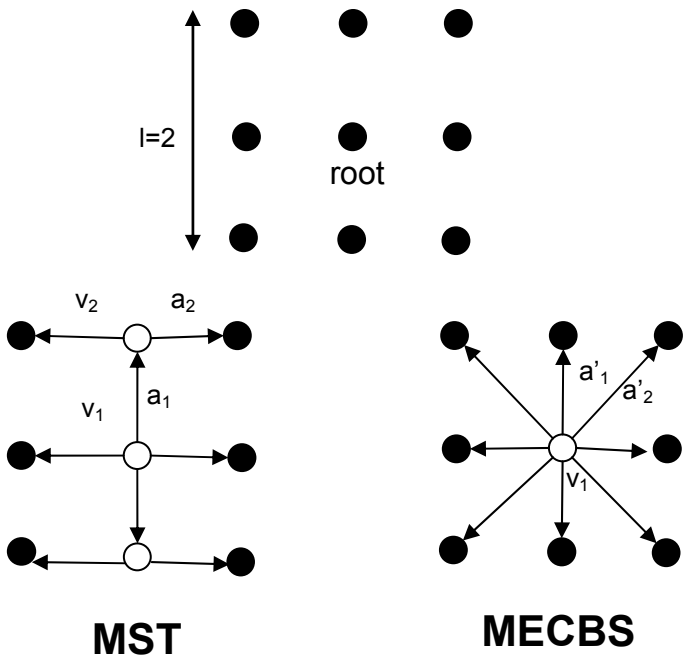


Fig 4.2 MST y MECBS: sólo los nodos blancos han de enviar información

En la anterior figura se puede observar que todas las aristas del MST tienen peso igual a uno. En cambio, en el problema del MECBS hay cuatro aristas de peso unidad y cuatro aristas de peso igual a dos. Recuérdese que el peso de una arista equivale al cuadrado de la distancia de los dos nodos unidos por la arista, es decir, a la longitud al cuadrado de la arista. De esta manera, resulta evidente que la suma de los pesos de las aristas es menor en MST –donde todas suman 8– que en MECBS –donde suman 12.

Por otro lado, si en ambos casos sumamos el radio de cobertura de los nodos de la red, puede verse que el MST no ofrece la mejor solución del problema. En el caso de la solución de Kruskal tenemos ΣR_i^2=3 ya que para que la información llegue desde el nodo v_1 al resto de la red, es necesario hacer retransmisiones a través de los nodos representados en blanco. De este modo, al final se obtiene un total de 3 nodos activos con radio de cobertura igual a uno. Si se analiza el mismo problema desde el punto de vista del MECBS, puede asignarse un radio de cobertura mayor al nodo central –igual a raíz de dos, que es la distancia que lo separa del nodo más alejado– suficiente para llegar al resto de la red. En este caso, se aumenta el radio de cobertura de dicho nodo, pero en compensación ya no serán necesarias las retransmisiones desde los otros dos nodos para llegar a todos los puntos. Es decir, al aumentar uno de los radios de cobertura eliminamos la necesidad de emplear los radios de otros dos nodos. Por este motivo el coste total de la red disminuye a ΣR_i^2=2.

Dicho de otra manera, en MECBS se pueden 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 estaríamos obligados a contar las dos aristas como radios de nodos diferentes v_1 y v_2. La clave reside, por tanto, en que con un solo

radio se puede cubrir, en ocasiones, más de una arista en el problema del MECBS. Y lo que deseamos en este caso es minimizar la suma de radios, no la suma de aristas. De esta manera, en el problema del MECBS se pueden tener aristas de mayor peso pero una suma de radios menor –ya que con un radio mayor se puede cubrir más de una arista– y, por tanto, un menor consumo de energía. Es por esto que aunque Kruskal ofrece la solución óptima del MST, no puede hacerlo para el MECBS, de modo que finalmente se ha aplicado únicamente como método de referencia para compararlo con los resultados de simulated annealing y del algoritmo hormigas.

4.3. Algoritmo de Kruskal para MECGS

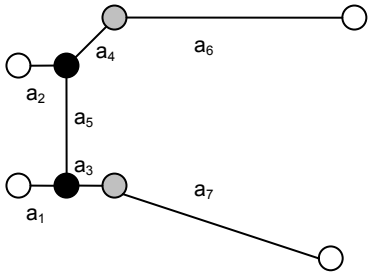
En el problema del MECGS todos los nodos han de enviar información al resto de la red. De este modo, cada uno de los puntos de la red no sólo mandará información al resto sino que además estará recibiendo datos de los *n-1* nodos restantes. En este caso, se ha obtenido en primer lugar, mediante Kruskal, un árbol generador mínimo MST. Como ahora el árbol se ha de recorrer en todas las direcciones, un nodo tendrá que alcanzar todos los nodos a los que está unido por el árbol (cosa que no sucedía en el caso del MECBS, ya que el árbol se recorría sólo en un cierto sentido que llevaba del nodo raíz al resto de nodos). Es por esto que a cada nodo se le ha de asignar un radio de cobertura igual a la longitud de la mayor de las aristas que inciden en él. Nótese que en el caso anterior del MECBS esto no era así ya que algunas aristas podían entrar y no salir en un nodo, de modo que no influían en su radio de cobertura.

Todo esto puede hacerse de la siguiente manera (más por cuestiones de análisis que por cuestiones prácticas, como se ve con más detalle en el apéndice de la presente memoria). En primer lugar, se ordenan las aristas del árbol generador mínimo MST de mayor a menor según su peso de modo que $a_{n-1} \geq a_{n-2} \geq \ldots \geq a_1$ (por simplicidad, asumimos que a_i es el peso de la arista a_i y que MST es la suma de todas esas aristas). De asta manera, la arista a_{n-1} se asignará a los dos nodos a los que es incidente (esto significa que dichos nodos tendrán un radio de cobertura igual a la longitud de a_{n-1}); a_{n-2} se asignará también a sus dos nodos incidentes (a no ser que uno de ellos incida en a_{n-1} y tenga por tanto asignado ya un radio de cobertura a_{n-1} mayor o igual al de a_{n-2}, en cuyo caso a_{n-2} sólo será asignado a un nodo); y lo mismo para las aristas siguientes, que podrán ser asignadas a sus dos nodos incidentes o a uno sólo (si uno de los nodos ya tiene asignado el radio de una arista anterior) o a ninguno (si sus dos nodos incidentes ya tienen asignado el radio). En la práctica, sin embargo, los radios de cobertura de los nodos pueden asignarse a medida que se construye el árbol. De esta forma, cada vez que Kruskal escoge una nueva arista, se asigna su longitud como radio de cobertura de sus dos nodos incidentes (anulando además la asignación de posibles radios anteriores, ya que se corresponderán con aristas de menor peso; recuérdese que Kruskal escoge las aristas de menor a mayor peso).

Como se ha dicho anteriormente, en el apéndice de la presente memoria se demuestra que, al igual que para MECBS, MST no ofrece la solución óptima en el caso del MECGS (y sin embargo, como se verá además en el apartado de resultados, puede afirmarse que el MST se aproxima más al MECGS que a MECBS). En la figura 4.3 se presenta un ejemplo en el que un árbol *A*, de mayor peso que MST, ofrece una solución mejor que MST para el MECGS. En las tablas se han enumerado las aristas de mayor a menor peso, se ha indicado dicho peso (calculado como la longitud al cuadrado de cada arista) y a continuación se ha indicado el número de nodos al que

ha sido asignado cada arista, es decir, el número de nodos cuyo radio de cobertura viene dado por cada una de las aristas (número que como se ha dicho es siempre dos para a_{n-1}; dos o uno para a_{n-2}; y dos, uno o cero para el resto de aristas).

Al sustituir en MST la arista que une los nodos representados en negro por la arista de mayor longitud que en *A* une los nodos representados en gris, obtendremos un árbol *A* de peso mayor que MST. Sin embargo, de esta forma se habrá eximido a los nodos negros de un radio igual al de la arista eliminada y los dos disminuirán su radio de cobertura. Por otro lado, aunque los nodos grises tengan una arista nueva, su radio no cambiará dado que son incidentes a aristas mayores que la nueva arista. Es por esto que finalmente *A* ofrece una mejor solución para el MECGS que MST, a pesar de tener un peso mayor. En suma, lo mismo que sucedía para el MECGS, el MST no siempre ofrece una solución óptima puesto que no pretendemos minimizar la suma de todas las aristas, sino la suma de los radios de cobertura de los nodos.



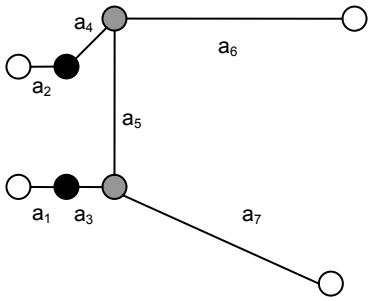
MST

	a_7	a_6	a_5	a_4	a_3	a_2	a_1
Peso	25	25	9	2	1	1	1
Nodos asignados	2	2	2	0	0	1	1

$\text{MECGS(MST)} = 120$

$\text{MST} = 64$

$\text{MST} + a_{n-1} = 89$



Árbol A

	a_7	a_6	a_5	a_4	a_3	a_2	a_1
Peso	25	25	16	2	1	1	1
Nodos asignados	2	2	0	1	1	1	1

$\text{MECGS(A)} = 105$

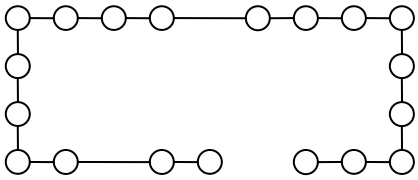
$\text{A} = 71$

$\text{MECGS(A)} < \text{MECGS(MST)}$

Fig 4.3 El árbol A da una mejor solución para el MECGS que el MST

En la figura 4.4 se muestra otro ejemplo en que el MST no ofrece la solución óptima del MECGS. En este caso la solución mejor que la de MST no se obtiene a través de otro árbol *A* sino de un ciclo. Por otro lado, en el apéndice se ha demostrado que para cualquier grafo la solución óptima del MECGS cumple la desigualdad $\text{MECGS}_{opt} \geq \text{MST}+a_{n-1}$, en donde MST se refiere, como se ha dicho, a la suma del peso de todas las aristas del árbol generador mínimo, y en donde a_{n-1} es el peso de la mayor arista de MST. Igualmente, en el apéndice se dan ejemplos de grafos para los cuales el valor óptimo de MECGS cumple la igualdad anterior, y de otros cuyo valor óptimo es superior a la cota $\text{MST}+a_{n-1}$.

De este modo, en el presente proyecto se ha aplicado el algoritmo de Kruskal para MECGS únicamente como modelo de referencia, dado que de lo dicho puede deducirse que MECGS(MST) ≥ $MECGS_{opt}$ ≥ $MST+a_{n-1}$. Igualmente, la solución general del problema se ha tratado de obtener de forma aproximada con los mismos algoritmos probabilistas presentados para MECBS.

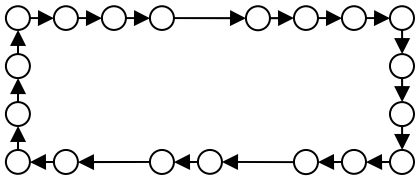


a) MST

$$MECGS(MST) = 15\cdot 1^2 + 4\cdot 2^2 = 31$$

$$MST = 16\cdot 1^2 + 3\cdot 2^2 = 28$$

$$MST + a_n = 28$$



b) MECGS

$$MECGS = 16\cdot 1^2 + 3\cdot 2^2 = 28$$

Fig 4.4 MECGS ≠ MST

4.4. Ordenación de los nodos

Tal como se ha explicado, los problemas planteados en el presente proyecto tienen como objetivo la asignación de radios de cobertura de los nodos de una red ad-hoc con el propósito de cubrir toda la red y, además, minimizar el consumo de energía. Supóngase, en este contexto, que se ha asignado al nodo *i* un radio R_i de tal modo que este nodo cubre los nodos $i_0,i_1,...,i_r$, ordenados de forma 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 que R_i es mayor que $d(i,i_r)$. En este caso, se pueden cubrir 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 permite cubrir los mismos nodos y reducir además el consumo de energía. De este modo, finalmente se ha asignado 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 este 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, al principio de hormigas y de simulated annealing se ha descrito, para cada nodo *i*, una lista con el resto de nodos ordenados de forma creciente según su distancia 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 cubiertos por *i*, conjunto formado por i_r más todos los nodos que en la lista de *i* aparecen antes que i_r.

4.5. Simulated annealing

Tal y como se señaló en el capítulo anterior, el algoritmo de simulated anealing se basa en la analogía del estado de un sistema físico de partículas y el espacio de soluciones de un problema de optimización combinatoria. En este contexto, la distribución de las moléculas del sistema puede hacerse corresponder con el espacio

de soluciones del problema, mientras que la energía del sistema puede equivaler a la función de coste del problema. La temperatura puede entenderse además 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.

Así, dado un conjunto *S* de nodos dentro de un cuadrado *C*, el algoritmo de simulated annealing asigna en primer lugar a cada nodo un radio de cobertura de manera que haya conectividad (de un nodo al resto en el caso del MECBS; de todos los nodos al resto en el caso del MECGS). A continuación el algoritmo escoge, en cada iteración, un nodo de forma equiprobable y le cambia su radio de cobertura o, mejor dicho, su rango. En nuestro caso, con una probabilidad P_e el algoritmo disminuye en una unidad el rango del nodo. Si el nuevo rango proporciona conectividad es aceptado. En otro caso, se recupera el rango anterior. Del mismo modo, con una probabilidad $1-P_e$ el algoritmo aumenta el radio de cobertura en una suma escogida aleatoriamente. En este caso, se conserva siempre la conectividad pero a la vez se aumenta la función de coste. De este modo, el nuevo rango es aceptado con una probabilidad $e^{-\Delta f/T}$, en donde Δf es la diferencia entre la nueva y la antigua función de coste y T es el parámetro de control llamado 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 la nueva solución.

El algoritmo repite la misma operación un número determinado de veces para cada temperatura: cada vez se escoge aleatoriamente un nuevo nodo y se le asigna un nuevo radio de cobertura según el procedimiento descrito. Una vez concluida la última iteración para una temperatura dada, esta se reduce multiplicándola por un factor menor a la unidad pero próximo a ella. 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.

Para ambos problemas –MECBS y MECGS– simulated annealing funciona de la misma manera, con la diferencia de que en cada caso se aplica un algoritmo distinto de comprobación de la conectividad. En el primer caso, debe garantizarse que el nodo raíz pueda mandar información al resto de la red, mientras que en el segundo caso todos los nodos tienen que ser visibles a todos los nodos.

4.6. Algoritmo de Hormigas

El algoritmo de hormigas, según se ha descrito en el anterior capítulo, consta de una serie de agentes que trabajan en paralelo para resolver un problema de optimización combinatoria. Inicialmente, el algoritmo asigna un radio a cada nodo de la red de manera que exista conectividad. Hecho esto, se colocan las hormigas aleatoriamente en la red. A continuación, en cada iteración cada hormiga se desplaza con probabilidad P_n al peor nodo vecino, es decir, a aquel que tiene un radio mayor y que consume por tanto mayor energía. En el caso de no superarse la prueba de probabilidad, la hormiga se desplaza a cualquier otro nodo de la red escogido de forma equiprobable. Una vez desplazada al nuevo nodo, la hormiga reduce con una probabilidad P_b el rango del nodo la máxima cantidad posible de modo que se siga cumpliendo la condición de conectividad. En otro caso (i.e. con una probabilidad 1- P_b) la hormiga aumenta en una unidad el rango del nodo. Como siempre, se introduce la posibilidad de un empeoramiento eventual con el fin de evitar los extremos locales de la función de coste. El procedimiento se repite hasta que el algoritmo converge. Al

igual que en simulated annealing, el funcionamiento de hormigas es exactamente igual para cada uno de los problemas de MECBS y MECGS, con la salvedad de la parte del algoritmo que comprueba la conectividad.

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, debe ser sin embargo simple para no afectar 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.

CAPÍTULO 5. RESULTADOS

En el presente capítulo se presentan los resultados obtenidos tras aplicar los algoritmos de Kruskal, simulated annealing y hormigas, tanto al problema del subgrafo de difusión de energía mínima MECBS, como al subgrafo de gossiping de energía mínima MECGS (recuérdese que, como se ha dicho en los capítulos anteriores, la diferencia entre MECBS y MECGS estriba en que en el primero un solo nodo envía información al resto de la red, mientras que en el segundo todos los nodos deben mandar información a los demás.) Dado que el problema del MECBS ya ha sido analizado en proyectos anteriores [4,20], en el presente trabajo se ha tratado fundamentalmente de obtener mejoras en el comportamiento de los algoritmos y de ampliar su aplicación al caso del MECGS. De este modo, no sólo se van a ofrecer comparaciones obtenidas por los distintos algoritmos para un mismo problema (MECBS o MECGS), sino que igualmente se compararán los resultados obtenidos sobre las mismas redes para el caso de MECBS, por un lado, y para el caso de MECGS, por otro, con el objeto de determinar el coste energético de la transmisión de información desde todos los nodos al resto de la red.

5.1 Pruebas

Los algoritmos de kruskal, hormigas y simulated annealing han sido programados en lenguaje C estándar sobre una plataforma Linux. Los tres han sido programados para realizar pruebas sobre un cuadrado de lado unidad, para redes que varían su número de nodos entre 10 y 300. Los nodos han sido distribuidos aleatoriamente y de forma uniforme sobre el cuadrado de lado uno, de modo que los resultados obtenidos en cuadrados mayores serían equivalentes a los del presente proyecto, multiplicados en cada caso por un determinado factor de escala. Para cada valor n del número de nodos, se han creado cinco redes aleatorias distribuyendo los n puntos de forma equiprobable a lo largo de todo el cuadrado. Hay que tener en cuenta que, para que las comparaciones tengan sentido, es necesario que las cinco redes aleatorias sean las mismas para cada conjunto de n puntos y para todos los algoritmos.

En proyectos anteriores, al analizar el problema de subgrafo de difusión de energía mínima se realizaron pruebas con los tres algoritmos, los dos probabilistas ya explicados, más un algoritmo determinista basado en Kruskal. Como ya se ha explicado, aunque MECBS y MST no son el mismo problema, se ha tomado Kruskal como modelo de referencia. En este proyecto los resultados de Kruskal para MECBS se han tomado de los proyectos anteriores, ya que al tratarse de un algoritmo determinista y trabajar sobre las mismas redes, no es posible mejorar los resultados. De este modo sólo hemos programado Kruskal para el caso del MECGS.

En el caso de simulated annealing se han realizado las pruebas para el caso de $9 \cdot 10^4$ iteraciones por temperatura, ya que en anteriores pruebas los resultados con más iteraciones apenas mejoraban la función de coste y por otro lado aumentaban significativamente el tiempo de cálculo. Los parámetros utilizados T_o, T_f, R y P_e para las pruebas se han tomado asimismo de los proyectos anteriores [4,20]. T_o corresponde a la temperatura inicial con la que empieza a funcionar el algoritmo, T_f es la temperatura final en la que acaba de ejecutarse el algoritmo, R corresponde al factor de enfriamiento y P_e es la probabilidad de permitir empeoramientos con la finalidad de evitar los mínimos locales, tal y como se ha descrito el algoritmo en el anterior capítulo.

En la práctica, P_e es un valor menor a la unidad que en cada iteración se compara con la función del sistema "rand1" que genera números aleatorios y de forma equiprobable entre 0 y 1. En el caso en que la función devuelve un valor por encima de P_e, el algoritmo reduce el rango del nodo en una unidad, para así disminuir su radio de cobertura y, en consecuencia, la función de coste de la red. Esta mejora, obviamente, es aceptada sólo cuando la disminución del rango no rompe la conectividad del grafo (en otro caso se vuelve al rango inicial del nodo). En el caso en que la función "rand1" no supera el umbral de P_e, el algoritmo aumenta el rango del nodo en una unidad con la finalidad de permitir empeoramientos y no caer en mínimos locales. Este empeoramiento se acepta con una probabilidad determinada por la función de Boltzman $e^{-\Delta f/T}$.

La selección de parámetros se ha realizado siguiendo criterios estadísticos. En el caso de las temperaturas, tanto la inicial como la final, se ha tenido en cuenta que la probabilidad de obtener una solución peor que la anterior se calcula como $e^{-\Delta f/T}$. Si, por ejemplo, situamos la temperatura inicial en un valor de 0.1, la probabilidad de que el radio al cuadrado de un nodo empeore en un valor de 0.01 se calcula como $e^{-0.01/0.1} = 0.90$, resultado que tiene sentido ya que al principio del algoritmo se permiten ciertos empeoramientos con el objetivo de evitar los mínimos locales de la función de coste. Por otro lado, al final del algoritmo la probabilidad de aceptar soluciones peores ha de ser casi nula. Por esta razón, a medida que nos acercamos a la temperatura final, la probabilidad de aceptar un empeoramiento ha de disminuir. De esta manera, si la temperatura final es de 10^{-4}, la probabilidad de aceptar un empeoramiento de 0.01 es de $e^{-0.01/0.0001} = 3.72 \cdot 10^{-44}$, cantidad despreciable si se tiene en cuenta el número de iteraciones que efectúa el algoritmo por cada temperatura.

El factor de enfriamiento R, que corresponde al valor por el que se multiplica la temperatura al llegar al final de las iteraciones correspondientes, ha de ser cercano a la unidad. Cuanto más se acerca a la unidad, más lentamente disminuye la temperatura, y por tanto más valores tenemos para hacer pruebas. Si tomamos como temperatura inicial y final las expuestas en el ejemplo anterior, 10^{-1} y 10^{-4} respectivamente, y tomamos como factor R de enfriamiento el valor de 0.99, el algoritmo realizará un total de 688 enfriamientos ya que $10^{-4} \approx 1 \cdot 10^{-1} \cdot 0.99^{688}$. De esta manera se obtiene que el número total de iteraciones es de $688 \cdot 9 \cdot 10^4 = 61{,}92 \cdot 10^6$. En la tabla 5.1 se han añadido los valores de los parámetros escogidos, según los resultados de los proyectos anteriores.

En el caso del algoritmo de hormigas también se han tomado los parámetros de los proyectos anteriores: P_{best}, P_{node}, P_{earth}, y el número de hormigas. P_{best} es la probabilidad de asignar a un nodo el menor radio de cobertura posible, siempre y cuando se conserve la conectividad (en caso contrario, el algoritmo aumenta el radio de forma aleatoria para evitar mínimos locales). Asimismo, P_{node} corresponde a la probabilidad de que una hormiga vaya al peor nodo adyacente (en el caso contrario, se desplaza a un nodo escogido al azar). P_{earth} es la probabilidad de mover todas las hormigas de forma aleatoria. El algoritmo se para una vez realizadas un total de $12 \cdot 10^6$ iteraciones. Aunque parece ser una cifra muy distante a las $61{,}92 \cdot 10^6$ iteraciones de simulated annealing, pueden considerarse equivalentes puesto que hormigas realiza más operaciones que simulated annealing en cada iteración.

Como se explicó en el capítulo anterior, el algoritmo de hormigas ha sido programado de manera que en cada iteración una de las hormigas –escogida aleatoriamente– se desplaza desde su posición actual al peor de los nodos adyacentes, es decir, hasta el nodo adyacente que consume una cantidad mayor de energía. Como se verá más adelante, existe una variación del algoritmo que desplaza la hormiga al peor nodo de

toda la red. Esta operación se realiza con una probabilidad P_{node} que se compara con un número generado por la función “rand1”. En otro caso, la hormiga se desplaza a cualquier otro nodo del grafo, escogido de forma equiprobable. Acto seguido, si una hormiga supera la probabilidad P_{best}, es decir, si la función “rand1” obtiene un número por debajo de P_{best}, intenta mejorar el radio de cobertura del nodo también de dos posibles maneras dependiendo de la modalidad del algoritmo: todo lo posible mientras haya cobertura o en una unidad. Además, ocasionalmente entra en juego la probabilidad P_{earth} según la cual se reubican de forma aleatoria todas las hormigas del algoritmo, con el fin de evitar mínimos locales. Los valores de los parámetros empleados en hormigas se recogen en la tabla 5.2.

Como queda dicho, para cada valor *n* del número de nodos, se han creado cinco redes aleatorias diferentes. Sobre cada uno de los cinco grafos se han aplicado los algoritmos probabilistas cinco veces, obteniendo, de esta manera, un total de veinticinco resultados diferentes para cada valor de *n*. Por otro lado, como Kruskal es un algoritmo determinista se ha aplicado una sola vez sobre cada grafo.

5.2 Resultados MECBS

A lo largo de este apartado se presentan las diferentes mejoras aplicadas sobre los algoritmos probabilistas de simulated annealing y hormigas y los resultados obtenidos para cada caso. El valor que se presenta en cada caso corresponde a la media aritmética del mejor resultado de cada una de las redes. Es decir, para cada red se ha aplicado cada algoritmo cinco veces. Como se trata de algoritmos probabilistas, se han obtenido cinco resultados distintos de los cuales hemos tomado el mejor. Por último, los resultados presentados en las tablas se han obtenido calculando la media aritmética de los mejores resultados de cada una de las cinco redes (todo esto dado un determinado número de nodos). El coste de la red es, según se ha explicado en capítulos anteriores, la suma del cuadrado de los radios de cobertura de todos los nodos de la red.

5.2.1 Resultados para simulated annealing

En anteriores proyectos [4,20], como se ha explicado, el algoritmo de simulated annealing sólo podía reducir en una unidad el rango de un nodo cuando se conservaba la conectividad del grafo. En el presente proyecto, no obstante, con el fin de ampliar la exploración del espacio de soluciones se han permitido eventuales secuencias de rangos en los que el grafo resultante no tiene conectividad. Por esto se ha propuesto aceptar, bajo una probabilidad llamada P_{span} de valor muy pequeño, secuencias de rangos del problema que no proporcionan conectividad (secuencias a las que por simplicidad también llamamos soluciones, aunque no sean válidas). Evidentemente, en aquellos casos en los que se han aceptado soluciones sin conectividad, aunque se ha almacenado la función de coste para obtener con mayor facilidad el coste de las siguientes soluciones, dicha función no ha sido aceptada en caso de ser más baja que la mejor solución obtenida hasta entonces, puesto que se trata como se ha dicho de una solución no válida.

De este modo, aunque se ha permitido la posibilidad de explorar soluciones sin conectividad, estas soluciones se han mantenido por lo general próximas a otras soluciones que ofrecen conectividad, dados los valores muy próximos a cero que se han asignado a P_{span}. Por otro lado, para ajustar el valor de P_{span} según el orden de la

red, se han realizado una serie de pruebas. Todo ello ha permitido, en suma, obtener mejoras para las redes de 10, 50, 75, 100 y 200 nodos, sombreadas en la tabla en color gris. Esto es así porque, como se ha dicho, la posibilidad eventual de aceptar soluciones sin conectividad mejora la riqueza de la exploración del espacio de soluciones, aunque de forma algo aleatoria como se desprende de la variación de los resultados según el número de nodos de la red.

n	Coste anterior	Coste red	Tiempo anterior	Tiempo ejecución	T_o	T_f	R	P_e	P_{span}
10	0.405521	0.40541	291.4	600.929	0.2	10^{-3}	0.99	0.5	$5 \cdot 10^{-6}$
25	0.398907	0.398907	782.9	1428.51	0.2	10^{-4}	0.99	0.5	0
50	0.372589	0.37175	1153	1870.08	0.1	10^{-3}	0.99	0.5	$10 \cdot 10^{-6}$
75	0.371592	0.369439	1028	2189.88	0.01	10^{-4}	0.99	0.5	$75 \cdot 10^{-6}$
100	0.376477	0.375372	1431	2917.87	0.01	10^{-4}	0.99	0.5	$10 \cdot 10^{-6}$
150	0.388545	0.388545	288	477.057	0.2	10^{-4}	0.75	0.5	0
200	0.362209	0.361345	12368	19000.3	0.2	10^{-4}	0.99	0.5	$5 \cdot 10^{-6}$
300	0.369597	0.369597	18750	26009.2	0.1	10^{-4}	0.99	0.5	0

Tabla 5.1 Resultados del algoritmo simulated annealing. Los tiempos se expresan en segundos de CPU.

5.2.2 Resultados para hormigas

Como se ha explicado en el capítulo anterior, la diferencia entre hormigas y simulated annealing reside en el hecho de que hormigas realiza una búsqueda dirigida en el espacio de soluciones. De este modo, el algoritmo hormigas, a diferencia de simulated annealing, no escoge aleatoriamente un nodo antes de cambiar su radio de cobertura, sino que esta elección sigue un criterio: como se ha dicho, hormigas escoge el peor nodo adyacente al nodo en el que se encuentra cada hormiga.

En proyectos anteriores se plantearon distintas variaciones sobre el movimiento de las hormigas en la red. Por un lado, una hormiga puede desplazarse al peor nodo adyacente. Por otro lado, existe la posibilidad de que cada hormiga se desplace no a su peor nodo adyacente, sino al peor nodo de toda la red. Ambos movimientos fueron probados y analizados en [4], obteniendo mejores resultados el segundo caso. La explicación del mejor funcionamiento del segundo movimiento de las hormigas es que, suponiendo que en una mayoría de casos se reduce el rango del nodo, ya que P_{best} es un valor próximo a la unidad, los vecinos del nodo se quedan en una situación delicada de conectividad, ya que un nodo cercano tiene en esos momentos un radio de cobertura más pequeño. Esto da poca flexibilidad a la hora de disminuir el rango de los nodos adyacentes. Dicho de otra manera, si una vez reducido el rango de un nodo se trata a continuación de disminuir el rango de un nodo vecino, seguramente no se podrá en la mayoría de los casos ya que la conectividad del grafo estará bastante amenazada (por el hecho de haberse recortado recientemente el rango de un nodo adyacente). Esto resta eficiencia al algoritmo, de modo que en general van a obtenerse mejores soluciones en el caso en el que la hormiga se desplaza no al peor nodo adyacente sino al peor nodo de toda la red. En este caso, con todo, aumenta el coste computacional ya que cuesta más evaluar el estado de toda la red (para seleccionar el peor nodo) que el conjunto restringido de los nodos adyacentes al nodo actual. Por otro lado, en este caso deja de tener sentido el parámetro del número de hormigas, puesto que todas las hormigas van a tener visibilidad sobre toda la red (no

confinarán su búsqueda a zonas restringidas del grafo) y en consecuencia tanto será tener una única hormiga como un conjunto numeroso, puesto que todas realizan la misma operación.

Existe además la posibilidad de mejorar el rango de un nodo seleccionado no como se ha hecho hasta ahora, en una cantidad máxima siempre y cuando se conserve la conectividad del grafo, sino de unidad en unidad, al igual que en simulated annealing. Esto se hace así para uniformizar o distribuir entre distintos nodos la "tensión" (en la conectividad) provocada por las sucesivas disminuciones de los radios de cobertura y, en suma, para flexibilizar la búsqueda en el espacio de soluciones. Entonces, se define un algoritmo que actúa de la manera siguiente: antes de desplazar la hormiga de un nodo a otro, se comprueba el coste de la solución actual de la red. Si este valor es inferior a un determinado valor C_u se desplaza la hormiga hacia el peor nodo adyacente y a este nodo se le disminuye una unidad el rango con probabilidad P_{best}, siempre y cuando siga existiendo conectividad (en otro caso, el rango se aumenta en una unidad). En cambio, si el valor del coste es mayor o igual a C_u la hormiga se desplaza hacia el peor nodo de la red y entonces se disminuye su rango en la máxima cantidad posible (y no sólo en una unidad, como en el caso anterior), otra vez con probabilidad P_{best}. En los dos casos, además (tanto si la función de coste es menor a C_u como si la supera) el desplazamiento al peor nodo (adyacente o de toda la red, según el caso) se realiza bajo una determinada probabilidad P_{node}. En otro caso, la hormiga se desplaza a cualquier otro nodo de la red escogido de forma equiprobable.

Lo que se intenta con esta modificación es lo siguiente. En aquellos casos en que la solución está alejada de la solución óptima (y tiene por tanto un coste superior a C_u) se trata de que el algoritmo converja con más rapidez (por eso la disminución del rango es la mayor posible en cada iteración). Por otro lado, en aquellas ocasiones en que la solución empieza a aproximarse a una posible solución óptima (y está por tanto por debajo de C_u) se trata de flexibilizar la exploración del espacio de soluciones, reduciendo el rango de unidad en unidad.

Al ejecutar esta modalidad del algoritmo de hormigas en el anterior proyecto [4] se obtuvieron mejoras en todos los casos. Por este motivo, en el presente proyecto se ha partido de este modelo híbrido. En un principio, no obstante, se ha tratado de invertir el funcionamiento del algoritmo. Así, cuando el resultado está por encima del umbral de C_u, se ha disminuido en una unidad el rango y se ha desplazado la hormiga al peor nodo adyacente; y por debajo de este valor se ha tratado de mejorar todo lo posible el peor nodo de la red. Por otro lado, se ha tomado además como valor de comparación con C_u el del mejor valor calculado hasta entonces. De esta manera el algoritmo analiza el espacio de soluciones inicialmente de una manera más lenta, pero una vez se acerca a la solución óptima, se intenta conseguir que converja rápidamente hacia el final. Igualmente, se ha tratado de mejorar el algoritmo aplicando para ambos movimientos una reducción del rango de una unidad (tanto por encima como por debajo de C_u). En ninguno de estos casos, no obstante, se han conseguido mejorar los resultados del proyecto inicial. Finalmente, al igual que se ha hecho en simulated annealing, se han permitido eventualmente, bajo una probabilidad P_{span}, soluciones en las que el grafo resultante no tiene conectividad.

Como las hormigas se mueven de dos maneras diferentes según estén por encima o por debajo de un determinado coste umbral llamado C_u, debe calcularse en cada caso, en función del número de nodos de la red, dicho valor C_u de referencia. En nuestro caso hemos realizado una serie de pruebas sobre redes de 50 nodos, para diferentes valores de C_u, y se ha escogido el que ofrece mejores soluciones. A partir de ahí, para encontrar el valor de C_u apropiado para cada red de n nodos, se ha planteado una regla de tres multiplicando C_u por la misma proporción en que se encuentran las

mejores soluciones calculadas hasta entonces. Una vez establecido el umbral, existen cuatro combinaciones posibles para desplazar las hormigas.

- Desplazarse al peor nodo de la red, disminuir todo lo posible el rango, y aceptar condiciones de no conectividad.
- Desplazarse al peor nodo de la red, disminuir todo lo posible el rango, y no aceptar condiciones de no conectividad.
- Desplazarse al peor nodo adyacente, disminuir en una unidad el rango, y aceptar condiciones de no conectividad.
- Desplazarse al peor nodo adyacente, disminuir en una unidad el rango, y no aceptar condiciones de no conectividad.

Se han realizado pruebas combinando las opciones de movimiento de modo que, por encima del umbral C_u, la hormiga se desplaza al peor nodo de la red y por debajo de dicho umbral se desplaza al peor nodo adyacente (siempre con una probabilidad P_{node}). El hecho de aceptar errores de conectividad por encima o por debajo del umbral hace que el coste total de la red varíe. En concreto, si se aceptan errores tanto por encima como por debajo de C_u, el coste global aumenta, obteniendo peores resultados. A raíz de las diferentes ejecuciones realizadas, se llega a la conclusión de que el modelo más adecuado no debe aceptar errores de conectividad por encima de C_u, y sí debe aceptarlos por debajo de dicho valor. De esta manera, por debajo del valor de C_u, las hormigas se desplazan hacia el peor nodo adyacente y disminuyen el rango en una unidad, y en el caso de no existir conectividad, se acepta la nueva solución bajo una probabilidad P_{span}, todo con el objeto de dar flexibilidad a la exploración del espacio de soluciones conforme el algoritmo se acerca a la solución óptima. A continuación se muestra una tabla con los valores obtenidos para C_u y P_{span} en función del número de nodos de la red. Como en la tabla anterior, se han sombreado en gris los casos en los que se ha mejorado la función de coste final.

n	Coste anterior	Coste red	Tiempo anterior	Tiempo ejecución	P_{best}	P_{node}	P_{earth}	*Horm*	C_u	P_{span}
10	0.3991	0.3937	138.65	270.1	0.5	0.15	0.1	10	0.42	$5 \cdot 10^{-4}$
25	0.3998	0.3998	193.09	351.7	0.75	0.15	0.1	1	0.42	0
50	0.4030	0.4019	355.53	577.1	0.85	0.15	0	1	0.42	$5 \cdot 10^{-4}$
75	0.3985	0.3985	586.69	831.1	0.85	0.15	0	5	0.42	0
100	0.4311	0.4311	664.07	1072.4	0.95	0.15	0	10	0.45	0
150	0.4696	0.4696	1951.1	2285.8	0.95	0.85	0.1	3	0.5	0
200	0.4501	0.4360	2165.9	2882.7	0.95	0.5	0	1	0.475	10^{-3}
300	0.4588	0.4495	4187	5163.4	0.95	0.85	0.1	3	0.485	$5 \cdot 10^{-4}$

Tabla 5.2 Resultados del algoritmo de hormigas

5.2.3 Comparativa

Para finalizar el estudio del subgrafo de energía mínima MECBS, en la figura 5.3 se ofrece una comparación de los resultados obtenidos con ambos algoritmos. Tal y como se ha visto en los anteriores proyectos, simulated annealing obtiene mejores resultados para las redes con mayor número de nodos y hormigas para las redes más pequeñas. Las mejoras obtenidas en el presente proyecto, además, han acentuado las diferencias entre ambos algoritmos. Así, para las redes de 10 nodos, se ha reducido en ambos casos la función de coste al permitir soluciones sin conectividad con

probabilidad P_{span}, pero tal y como sucedía en las anteriores pruebas, hormigas ofrece todavía mejores resultados.

Para las redes de mayor número de nodos, sin embargo, simulated annealing presenta un mejor comportamiento. Esto es así puesto 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 ha de 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 aumenta con el número de nodos de la red. De este modo, al crecer el tamaño de la redes, hormigas ha de realizar cada vez un mayor número de operaciones y por eso se aleja de los resultados de simulated annealing. En nuestro caso, además, simulated annealing ha disfrutado de un mayor tiempo de ejecución, según se observa en la figura 5.4, pero cuando se ha tratado de aumentar el tiempo del algoritmo hormigas, apenas se han obtenido mejores resultados.

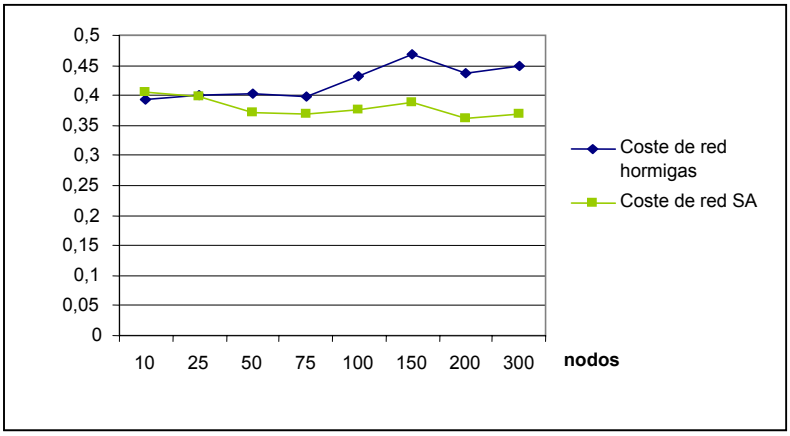


Fig 5.3 Comparativa función de coste

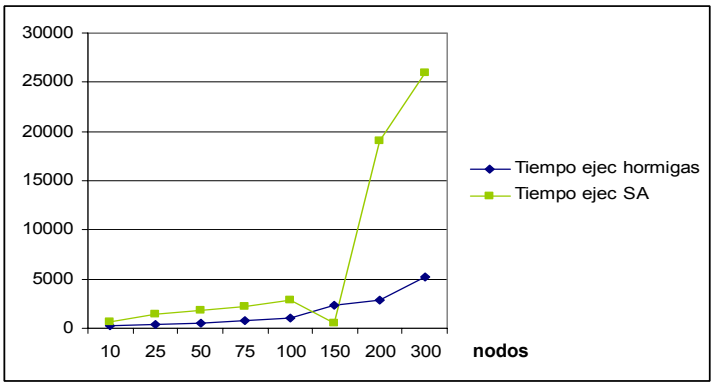


Fig 5.4 Comparativa tiempos de ejecución en segundos de CPU

Por otro lado, cabe señalar que la solución final se mantiene casi constante para los dos algoritmos, en función del número de nodos. Es decir, aunque al aumentar el número de nodos tendremos que mandar información a un mayor número de destinatarios, por otro lado aumentará 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 figura 5.3, 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.

Para comparar el comportamiento de los algoritmos, la figura 5.5 muestra la estadística del rango final de los nodos para las redes de 50 nodos (en otros casos las gráficas son equivalentes). Recuérdese que el rango de un nodo se define como el número de nodos a los que alcanza finalmente con su radio de cobertura, es decir, el grado del nodo en el grafo resultante. Los resultados mostrados en la gráfica se han vuelto a promediar para veinticinco casos en el caso de simulated annealing y hormigas (cinco soluciones distintas obtenidas sobre cinco redes de cincuenta nodos) y para cinco en el de Kruskal (una solución para cada red). Como puede verse, el comportamiento es prácticamente idéntico para simulated annealing y hormigas, incluso después de haberse implementado las modificaciones descritas en este capítulo. Para las redes de 50 nodos, 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, casi no existen nodos con un rango mayor o igual a diez, de modo que tenemos 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.

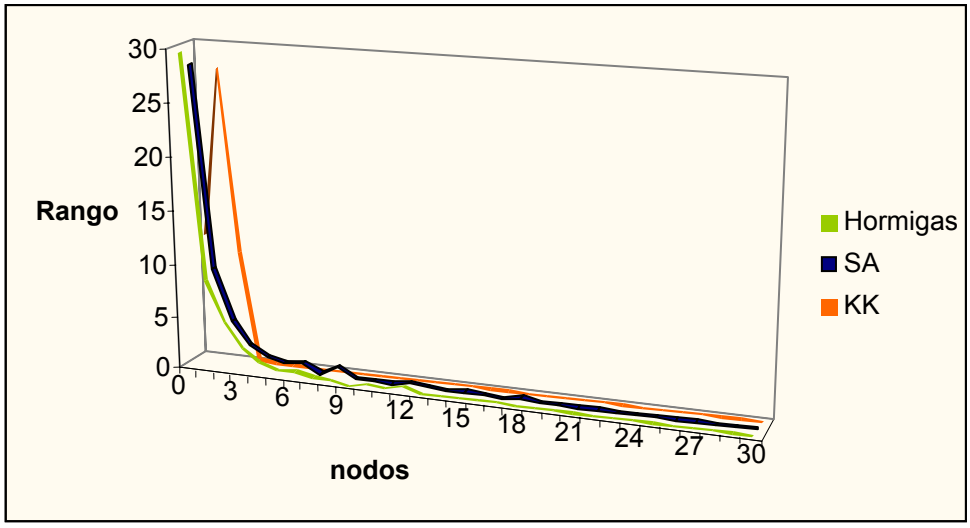


Fig 5.5 Estadísticas del rango para redes de 50 nodos

Igualmente, la figura muestra los rangos para las soluciones obtenidas por Kruskal en anteriores proyectos [4,20]. En este caso, y a diferencia de hormigas y simulated annealing, no encontramos un volumen tan grande de nodos con rango igual a 0 (15 para redes de 50 nodos, frente a los 30 de simulated annealing y hormigas). El pico de la función que representa las estadísticas de los rangos aplicando Kruskal se encuentra en un valor de rango igual a 1 (25 nodos de rango 1 para el caso de una red de 50 nodos). Todo esto se debe a la diferente naturaleza del algoritmo de Kruskal, que en primer lugar encuentra un árbol generador mínimo en el cual, según se ha visto

en el capítulo anterior, la mayoría de nodos están conectados como mínimo con algún otro nodo. Es por esto que no tendremos casi nodos de rango 0 (o como mínimo tendremos menos que por los casos de hormigas y simulated annealing). De hecho, en la figura 4.2 ya habíamos visto un ejemplo de todo esto. En ella se ve que en la solución encontrada por Kruskal han de emitir más nodos (3 en este caso) que en la solución encontrada por el problema del MECBS, en la cual sólo un nodo ha de enviar información. Igualmente, en aquel caso, se obtenían más nodos de rango 0 resolviendo directamente el problema del MECBS (8 nodos de rango 0) que aplicando primero Kruskal (6 nodos de rango 0).

n	MST+a(n-1)	Kruskal	SA	Hormigas
10	0.771544	0.831593	0.798589	0.798719
25	0.741765	0.918329	0.784885	0.785937
50	0.634414	0.816169	0.703998	0.716609
75	0.606595	0.799548	0.670433	0.697809
100	0.602095	0.808982	0.667611	0.684728
150	0.605251	0.814248	0.684676	0.718793
200	0.571223	0.767648	0.628202	0.681783

Tabla 5.6 Resultados para MECGS

La tabla 5.6 muestra los resultados obtenidos. Como en el caso del MECBS, para cada valor de *n* los valores representados corresponden a la media aritmética de los mejores resultados obtenidos para cada uno de los cincos grafos generados. Como se ha dicho, Kruskal no ofrece en general la solución óptima. Igualmente, simulated annealing y hormigas ofrecen mejores resultados que Kruskal y, en general, simulated annealing ofrece ligeramente mejores soluciones que hormigas, siempre por encima de la cota $MECGS|_{ópt} \geq a_{n-1} + MST$.

A medida que aumenta el número de nodos, los resultados de MECGS se alejan de la cota. Si se analizan los resultados obtenidos para cada grafo, dentro de un mismo número de nodos, existen resultados que se acercan mucho más a la cota que otros, dependiendo de la topología de la red. La evolución de los resultados en función del número de nodos se puede apreciar en la figura 5.7, donde se realiza una comparativa entre la cota y el coste óptimo calculado por ambos algoritmos. En este caso, en vez de promediar las mejores soluciones de todas las redes de un determinado número de nodos, se han puesto los resultados uno detrás de otro. Así, en primer lugar se han dibujado los cinco mejores resultados (para simulated annealing y hormigas) para las redes de 10 nodos, a continuación los cinco mejores resultados para las redes de 20 nodos y así sucesivamente. En cada caso se han comparado dichos valores con la cota a_{n-1} + MST obtenida por Kruskal.

Para las redes pequeñas de 10 nodos puede verse que en tres de los cinco casos los dos métodos probabilistas han obtenido la solución óptima, dada por a_{n-1} + MST. Esto es así puesto que se han obtenido soluciones idénticas a la cota a_{n-1} + MST. En los otros casos, las soluciones han sido superiores a dicha cota, lo cual no significa que necesariamente no sean óptimas (puesto que como se demuestra en el apéndice, existen topologías para las cuales $MECGS|_{ópt} > a_{n-1}$ + MST). En estos casos se podría comprobar mediante otros procedimientos (el más sencillo conceptualmente es el de búsqueda exhaustiva, que explora todo el espacio de soluciones) si la solución obtenida se corresponde con la solución óptima. Esto, en general, sólo es posible para redes pequeñas. Por otro lado, en ningún caso Kruskal ha obtenido la solución óptima, a pesar de lo cual las diferencias con simulated annealing y hormigas son menores que las que se obtienen en el caso del MECBS, según se ve más adelante en la tabla

5.9. De este modo, puede decirse que la complejidad del MECGS es algo menor que la del MECBS, puesto que se acerca más a las soluciones obtenidas por un algoritmo determinista de complejidad polinómica.

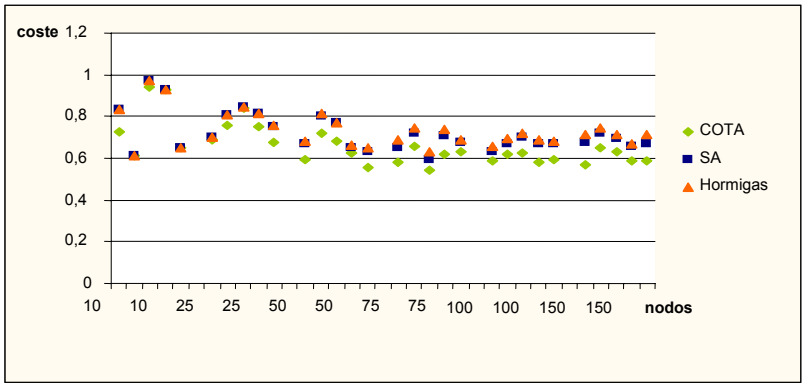


Fig 5.7 Soluciones de SA y hormigas vs. cota a_{n-1} + MST

5.3 Resultados para MECGS

En el presente apartado se analizan los resultados obtenidos para el problema del MECGS. En este caso se ha vuelto a emplear el algoritmo de Kruskal como modelo de comparación con hormigas y simulated annealing. Cabe recordar, tal y como se especificó en el anterior capítulo, que el funcionamiento de simulated annealing y hormigas es exactamente el mismo para MECBS y MECGS, con la excepción de la función que comprueba la conectividad del grafo resultante. Es por esto que para ambos algoritmos se han aplicado inicialmente los mejores parámetros de funcionamiento obtenidos en MECBS, según las tablas del apartado anterior. Obviamente, esto no es óbice para que en un futuro puedan buscarse nuevos parámetros más ajustados al funcionamiento del MECGS.

Según se ha explicado brevemente en el capítulo 4 y con más detalle en el apéndice, en el caso de MECGS el algoritmo de Kruskal tampoco ofrece una solución óptima del problema. Igualmente, en el apéndice se demuestra que ningún algoritmo puede obtener soluciones por debajo del valor de a_{n-1} + MST, en donde MST es la suma de los pesos de todas las aristas de un árbol generador mínimo y a_{n-1} el peso de la mayor arista de dicho árbol generador mínimo. De este modo, $MECGS|_{ópt} \geq a_{n-1}$ + MST, de forma que si un método encuentra una solución de valor a_{n-1} + MST, dicha solución ha de ser óptima puesto que, como se acaba de decir, no existen soluciones por debajo de dicho umbral.

Para el caso del MECGS se han empleado las mismas redes del apartado anterior, sobre las que se han aplicado los mismos procedimientos estadísticos (dado un número de nodos se han definido cinco redes, sobre las cuales se han ejecutado cinco veces cada uno de los algoritmos probabilistas, y una vez el algoritmo determinista de Kruskal). En este caso, el algoritmo de Kruskal es todavía más simple que en el caso del MECBS, puesto que cada vez que se añade una arista al árbol del MST, se puede asignar la longitud de dicha arista como radio de cobertura de sus dos nodos. En caso de que alguno de los nodos tenga asignado un radio de una arista anterior, el antiguo radio es substituido por el nuevo ya que Kruskal selecciona las aristas de menor a mayor peso, y la nueva arista va a ser por tanto más restrictiva que la anterior. Por otro lado, Kruskal permite obtener la cota a_{n-1} + MST de la solución óptima.

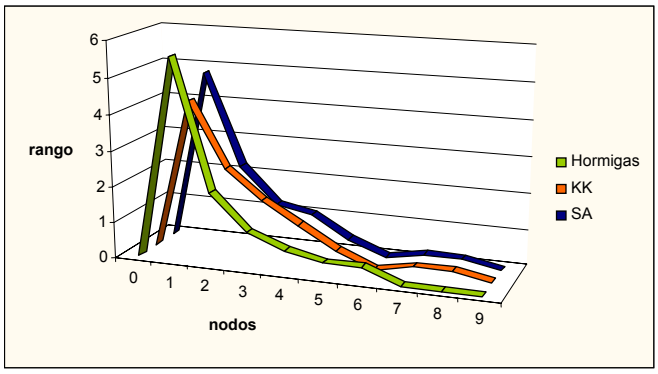


Fig 5.8 Estadísticas del rango para redes de 10 nodos

Finalmente, para comparar el comportamiento de los tres métodos, la figura 5.8 muestra la estadística del rango final de los nodos para las redes de 10 nodos. En este caso, y a diferencia de lo que sucede en las gráficas de la figura 5.5 para MECBS, no existen nodos de rango 0, ya que todos han de alcanzar como mínimo a un nodo vecino para desde allí dirigirse al resto de la red. De este modo, ahora existe una gran mayoría de nodos con rango 1, y por lo tanto en este caso del MECGS las estadísticas del rango de simulated annealing y hormigas se aproximan mucho más a las de Kruskal que en el caso del MECBS. Esto es una muestra más de que MECGS está más próximo a MST que MECBS.

5.4 Comparativa MECGS y MECBS

Para finalizar, en el presente apartado se comparan los resultados obtenidos para ambos problemas del MECBS y el MECGS, con el objeto de estimar el coste energético de mandar información desde todos los nodos a todos los nodos de la red (recuérdese que mientras que en MECBS un solo nodo raíz manda información al resto de la red, en MECGS todos los nodos intercambian información entre sí, y que en ambos casos se han trabajado con las mismas redes aleatorias). En la figura 5.9 se presentan los resultados obtenidos para cada caso por Kruskal, de un lado, y por simulated annealing y hormigas, por otro (en este segundo caso se ha escogido finalmente la mejor solución obtenida por uno de los dos métodos probabilistas).

	MECBS		MECGS	
n	Kruskal	SA/Horm	Kruskal	SA/Horm
10	0.481141	0.393676	0.831593	0.798589
25	0.5639	0.398907	0.918329	0.784885
50	0.479416	0.37175	0.816169	0.703998
75	0.507725	0.369439	0.799548	0.670433
100	0.49214	0.375372	0.808982	0.667611
150	0.507184	0.388545	0.814248	0.684676
200	0.476857	0.361345	0.767648	0.628202

Tabla 5.9 Comparativa algoritmos MECGS y MECBS

En primer lugar, cabe remarcar que en ambos casos los métodos de resolución probabilistas –simulated annealing y hormigas- obtienen mejores resultados que el método determinista de Kruskal. En el caso del MECBS estas diferencias pueden llegar incluso al 30%, mientras que en MECGS se reducen bastante. Esto se debe a que, como se ha dicho con anterioridad, MST se aproxima más a MECGS que a MECBS.

Por otro lado, lo mismo que sucedía en MECBS (y debido a los mismos motivos que ya se han explicado en el apartado 5.2.3) el coste de MECGS se mantiene prácticamente constante en función del número de nodos. Esto significa que conforme aumenta el tamaño de la red, la solución del problema del MECGS se hace proporcionalmente más eficiente, si se mide la eficiencia como la relación entre la energía total consumida y el número total de nodos que mandan información. Es decir, cuanto mayor sea el número de nodos de la red, menor será el consumo total de energía si se normaliza por el número total de nodos que mandan información. Como curiosidad, puede verse igualmente que el coste de MECGS es, en general, aproximadamente el doble que el coste de MECBS, con independencia del tamaño de la red.

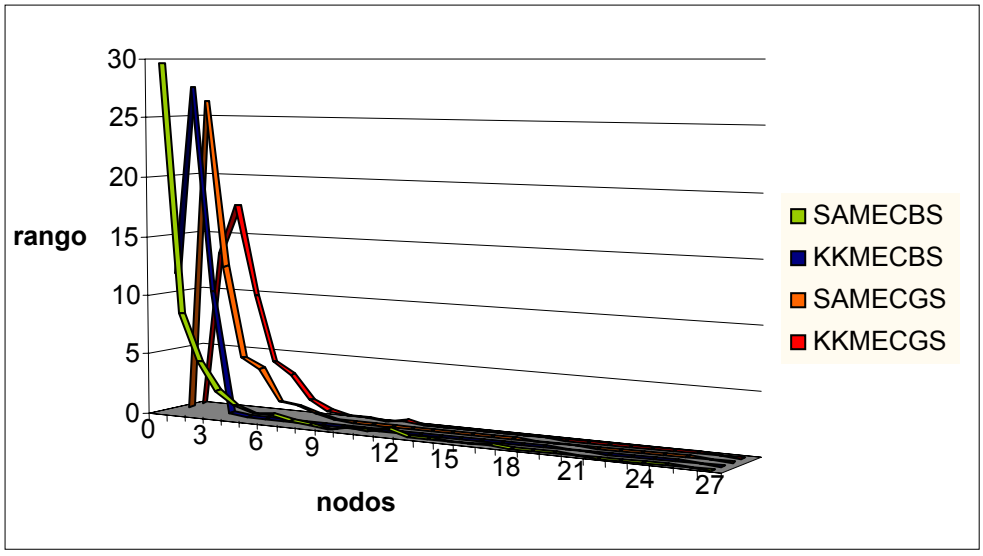


Fig 5.10 Grafica comparativa de rangos para redes de 50 nodos.

Finalmente, la gráfica 5.10 compara las estadísticas de los rangos de los nodos para simulated annealing (las de hormigas tienen la misma naturaleza y se han omitido por simplicidad) de un lado, y el algoritmo de Kruskal, de otro, en los dos escenarios de MECBS y MECGS. Según se ha explicado en anteriores apartados, en la figura se ve como las características de ambos algoritmos son mucho más próximas en el caso del MECGS que en el de MECBS.

5.5 Conclusiones

Como se ha visto a lo largo de la presente memoria, el problema del subgrafo de difusión de energía mínima MECBS es un problema NP-completo. Por otro lado, aunque no se ha demostrado que el problema del MECGS es NP-completo, sí se ha

visto que tampoco puede resolverse mediante técnicas deterministas simples basadas en el cálculo de árboles generadores mínimos. Esto quiere decir que en general habremos de aplicar técnicas de optimización combinatoria para su resolución y que los algoritmos deterministas no ofrecerán soluciones suficientemente buenas. Esto se ha visto en los apartados anteriores, donde los resultados obtenidos con Kruskal son peores que los que ofrecen los algoritmos probabilistas de simulated annealing y hormigas, tanto para el caso del MECBS como para el MECGS.

Por otro lado, el algoritmo de hormigas sigue comportándose mejor para redes pequeñas pero no se ha conseguido mejorar cuando el número de nodos aumenta, en cuyo caso simulated annealing obtiene mejores resultados (esto es así puesto que, como se ha dicho, hormigas es más sensible al incremento del número de nodos, es decir, debido a que el número de operaciones de hormigas crece con el número de nodos de forma muy acentuada, tanto para el MECBS como para el MECGS, cosa que no sucede en simulated annealing). En la práctica se puede aplicar un modelo combinado de tal forma que por debajo de un determinado número de nodos se aplique hormigas y por encima de este punto se utilice simulated annealing.

Igualmente, se han comparado ambos escenarios y se ha comprobado que cuando han de mandar información todos los nodos (MECGS) en vez de uno solo (MECBS), el consumo de energía se multiplica únicamente por un factor aproximado de 2, con independencia del número de nodos de la red. Se ha visto además que el algoritmo de Kruskal se acerca más a las soluciones ofrecidas por simulated annealing y hormigas en MECGS que en MECBS, lo que significa que la complejidad computacional de MECGS es ligeramente menor que la de MECBS.

Finalmente, cabe subrayar que el desarrollo de este proyecto puede tener un impacto medioambiental considerable, dado que en un futuro se ha de intensificar el uso de las redes ad-hoc, formadas en gran parte por dispositivos móviles. En este escenario, un estudio como el nuestro que permite optimizar el consumo total de baterías, podría traducirse en un futuro no sólo en un ahorro importante de energía sino también en una importante disminución de las toneladas de residuos –muchos de ellos tóxicos– que comporta el agotamiento de las baterías de los dispositivos electrónicos actuales. Cabe recordar, además, que este agotamiento se traduce muchas veces en la sustitución no solo de las baterías sino de todo el dispositivo portátil, como acostumbra a suceder con los teléfonos móviles, que tiramos a la basura cuando la batería deja de funcionar para comprarnos a continuación uno nuevo.

5.6 Líneas futuras

Aunque el presente proyecto amplia los resultados de proyectos anteriores [4,20], no deja de ser una primera tentativa de aplicar algoritmos heurísticos en el diseño de la topología de una red ad-hoc. Es por esto que existen numerosas líneas posibles de investigación.

En primer lugar, se puede tratar de mejorar los actuales algoritmos sin modificar la definición de los problemas planteados MECBS y MECGS, cambiando algunas de sus características o buscando con más detalle los mejores parámetros de funcionamiento, especialmente para el escenario del MECGS. En el caso del algoritmo hormigas, puede calcularse el valor de referencia C_u para cada grafo en función de la mejor solución encontrada hasta entonces (a diferencia de lo que hemos hecho hasta ahora en donde C_u toma el mismo valor para todos los grafos de un mismo número de nodos, con independencia de los resultados obtenidos para caso particular.) También

pueden aplicarse nuevos algoritmos heurísticos, 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 pueden 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. En el caso del MECGS, existe además la posibilidad de comprobar que algunas soluciones son óptimas, según se ha visto en apartados anteriores y se demuestra en el apéndice de la presente memoria, gracias a la cota $MECGS|_{\text{ópt}} \geq a_{n-1} + MST$. Queda pendiente, igualmente, comprobar (y demostrar en caso positivo) si MECGS es un problema NP-completo.

En segundo lugar, se puede modificar alguna característica de los problemas con el objetivo de representar con mayor exactitud la verdadera naturaleza de las redes ad-hoc. De este modo, en lugar de minimizar el sumatorio ΣR_i^2 puede tratar de minimizarse otro sumatorio ΣR_i^D donde el exponente $D \geq 2$ refleje con más fidelidad el consumo real de la energía de los nodos, que es proporcional a R_i^2 en el vacío, pero mayor en medios físicos reales. Se puede llegar a considerar por tanto la posibilidad de medir el exponente *D* de consumo sobre el terreno y utilizarlo en los cálculos de la función de coste de los algoritmos.

En el presente proyecto hemos supuesto que los nodos permanecen inmóviles, pero se puede plantear la posibilidad de que los nodos no ocupen una posición fija, sino que se vayan moviendo a lo largo del espacio. En este caso no solamente se tendría que contabilizar el consumo de energía de las diferentes soluciones, sino que también se tendría que tener en cuenta la velocidad de convergencia de los diferentes algoritmos. Esto se debe a que en un escenario variable se debe cambiar la topología de la red sobre la marcha, a medida que los nodos se van desplazando sobre el espacio, y por tanto necesitaremos algoritmos capaces de ofrecer soluciones óptimas o casi óptimas en intervalos de tiempo muy reducidos.

Por último, ya que el consumo de energía de un nodo se incrementa de forma considerable con el radio de cobertura, el algoritmo puede 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. Así, a los nodos con menor capacidad o con un tiempo de vida muy limitado por culpa de las baterías, se les pueden asignar radios de cobertura menores o directamente nulos de modo que vean aumentado su tiempo de vida. De este modo aumentará el tiempo de vida del conjunto de la red sin necesidad de recargar las baterías continuamente.

BIBLIOGRAFIA

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

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

[3] 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.

[4] E. Aymamí, A. Rodríguez, *Subgrafs de energia minima en xarxes ad-hoc,* Trabajo Final de Carrera de la Escuela Politécnica de Castelldefels (UPC), Castelldefels, 2008.

[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] I. Caragagiannis, Ch. Kaklamanis, P. Kanellopoulos, *Energy efficient wireless network Design*, Algorithms and Computation, vol. 2906, pp. 585-594, Springer Berlin/Heidelberg, 2003.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

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

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

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

[17] 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.

[18] 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.

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

[20] E. Ozón, *Subgrafos de Broadcasting de Energía Mínima*, Projecto de Final de Carrera de la Facultad de Informática de Barcelona (UPC), Barcelona, 2007.

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

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

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

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

APENDICE. SUBGRAFO DE GOSSIPING DE ENERGÍA MÍNIMA (MECGS)

En el presente apéndice se estudia el problema del del subgrafo de gossiping de energía mínima (MECGS según sus siglas en inglés: *Minimum Energy Consumption Gossiping Subgraph*) desde el punto de vista de la teoría de grafos. De este modo, se ofrecen métodos simples para su resolución y se ofrecen cotas para su solución óptima.

1. Definición de MECGS

Dado un cuadrado *C* de lado *l,* un conjunto *S* de *n* puntos o nodos distribuidos sobre *C*, el problema del MECGS consiste en asignar a cada nodo *i* de *S* un radio de cobertura R_i de tal modo que desde cualquier nodo pueda alcanzarse el resto de la red y que el sumatorio ΣR_i^2 sea mínimo. Como se ha visto a lo largo de la presente memoria, el problema del MECGS difiere del problema del subgrafo de difusión de energía mínima (MECBS) en la distribución de la información. De este modo, mientras que en MECBS un único nodo raíz ha de mandar información al resto de la red, en MECGS todos los nodos se han de comunicar con todos los nodos de la red.

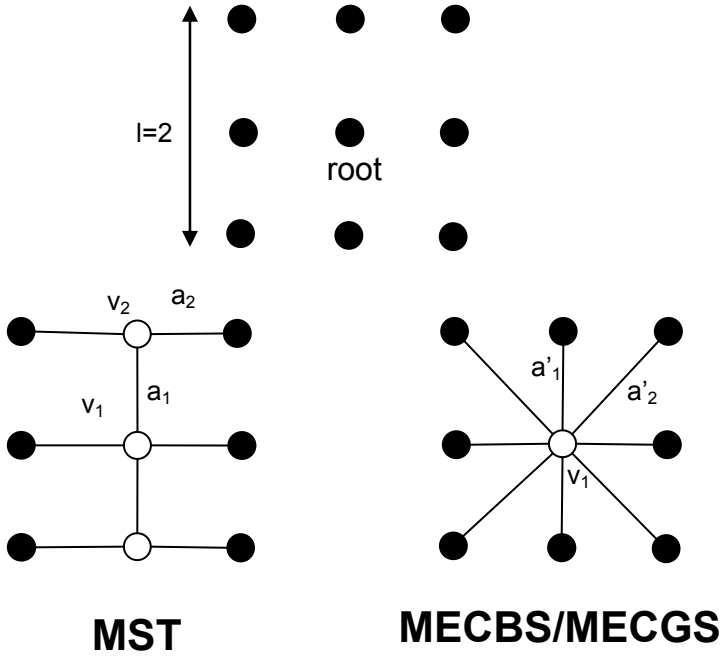


Fig. 1 MST vs. MECBS y MECGS

Así, en el cuarto capítulo hemos visto que para la red presentada en la figura 1, la segunda topología da una mejor solución para el problema del MECBS en el que el nodo raíz es el nodo central (la solución de la derecha da una función de coste de ΣR_i^2=2 frente a una función de coste de ΣR_i^2=3 de la solución de la izquierda, obtenida mediante el MST). En este caso, no obstante, si se plantea el problema del MECGS en el que todos los nodos han de mandar información al resto, la topología de la

izquierda, obtenida a partir del MST, da una función de coste igual a ΣR_i^2=9, mientras que la de la derecha da una solución considerablemente mayor ΣR_i^2=14. En este caso, además, es fácil comprobar que la solución ofrecida por MST es óptima. En efecto, en la solución planteada por MST todos los nodos tienen radio de cobertura igual a la unidad. Como además todos los nodos han de comunicarse como mínimo con algún nodo (para llegar al resto de la red a través de él) y como todos los nodos tienen su nodo más próximo a una distancia igual a uno, se deduce que ningún nodo puede tener radio menor que uno. Luego la solución obtenida es óptima, como se ha dicho. En todo caso, como se ve más adelante, el MST no va a ofrecer siempre la solución óptima del problema del MECGS (si bien siempre va a estar más cerca de dicho problema que del MECBS, tal y como se ha visto en el capítulo de resultados de la presente memoria).

No hay que olvidar, por otro lado, que si en MECGS se asigna un radio R_i a un nodo *i* tal que ese radio no coincide con la distancia de *i* a algún otro nodo, entonces la solución se puede mejorar inmediatamente ajustando el radio R_i a la distancia mayor de *i* a todos los nodos que alcanza con el radio R_i. De esta manera, aunque la conectividad no cambia se reduce el consumo total de energía ΣR_i^2 de modo que a lo largo del presente capítulo supondremos siempre que tenemos los radios ajustados.

2. Resolución de MECGS mediante MST

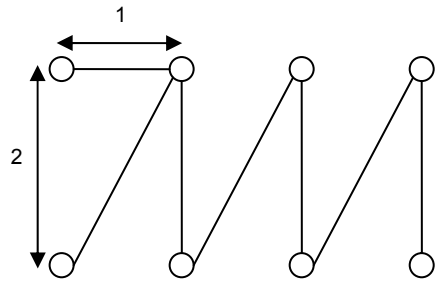
Como se ha explicado en el capítulo cuarto de la memoria, puede obtenerse una solución del problema del subgrafo de gossiping de energía mínima MECGS (solución que no va a ser siempre óptima), calculando un árbol generador mínimo MST del grafo mediante el algoritmo de Kruskal. En este caso, se ha considerado 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 un árbol generador mínimo MST mediante el algoritmo de Kruskal. En el problema del MECGS todos los nodos han de enviar información al resto de la red. De este modo, cada uno de los puntos de la red no sólo envía información al resto sino que además recibe datos de los *n-1* nodos restantes (de ahora en adelante supondremos que *n* es el orden del grafo). Esto significa que el árbol MST se ha de recorrer en todas las direcciones, de modo que un nodo tiene que alcanzar todos los nodos a los que está unido por el árbol (cosa que no sucede en el caso del MECBS, ya que el árbol se recorre sólo en un cierto sentido que lleva del nodo raíz al resto de nodos). De este modo, a cada nodo se le ha de asignar un radio de cobertura igual a la longitud de la mayor de las aristas que inciden en él en MST.

Todo esto puede hacerse de la siguiente manera (recuérdese, no obstante, que en la práctica se ha procedido de forma ligeramente distinta por razones de eficiencia, según se especifica en el cuarto capítulo). En primer lugar, se ordenan las aristas del árbol generador mínimo MST de mayor a menor peso de modo que $a_{n-1} \geq a_{n-2} \geq \ldots \geq a_1$ (por simplicidad, asumimos que a_i es el peso de la arista a_i y que MST es la suma de todas esas aristas en al árbol generador mínimo). De esta manera, la arista a_{n-1} se asignará a los dos nodos a los que es incidente (esto significa que dichos nodos tendrán un radio de cobertura igual a la longitud de a_{n-1}); a_{n-2} se asignará también a sus dos nodos incidentes (a no ser que uno de ellos incida en a_{n-1} y tenga por tanto

asignado ya un radio de cobertura a_{n-1} mayor o igual a a_{n-2}, en cuyo caso a_{n-2} sólo será asignada a un nodo); y lo mismo para las aristas siguientes, que podrán ser asignadas a sus dos nodos incidentes o a uno sólo (si uno de los nodos ya tiene asignado el radio de una arista anterior) o a ninguno (si sus dos nodos incidentes ya tienen asignado el radio).

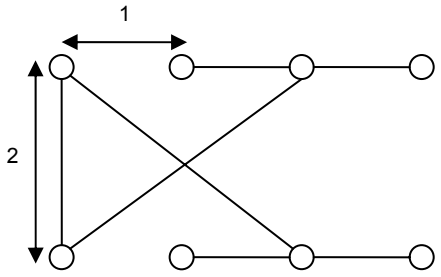
En la figura 2 se presentan un par de ejemplos de dicho procedimiento. En las tablas que acompañan a cada grafo se han enumerado las aristas de mayor a menor peso, se ha indicado dicho peso (calculado como la longitud al cuadrado de cada arista) y a continuación se ha indicado también el número de nodos al que ha sido asignado cada arista, es decir, el número de nodos cuyo radio de cobertura viene dado por cada una de las aristas (número que como se ha dicho es siempre dos para a_{n-1}; dos o uno para a_{n-2}; y dos, uno o cero para el resto de aristas). A lo largo de todo el apéndice emplearemos las mismas tablas. Igualmente, y por simplicidad, *A* podrá referirse tanto al árbol propiamente dicho como a la suma de los pesos de todas sus aristas; igualmente emplearemos la misma notación a_i para referirnos tanto a una arista como a su peso (aunque en ambos casos escribiremos el rótulo en cursiva –*A* o a_i– si nos referimos al árbol o a la arista, y en redonda –A o a_i– si nos referimos a los pesos). Igualmente, expresamos como MECGS(A) el valor de la solución obtenida a partir del árbol *A*, según el procedimiento de asignación descrito en el párrafo anterior.



	a_7	a_6	a_5	a_4	a_3	a_2	a_1
Peso	5	5	5	4	4	4	1
Nodos asignados	2	2	2	1	0	0	1

$$A = \sum a_i = 28$$

$$MECGS(A) = 35$$



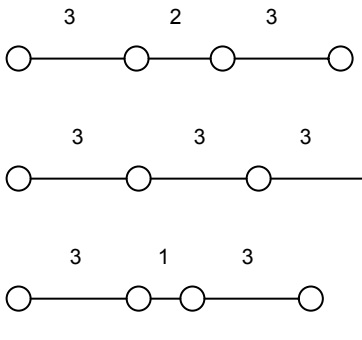
	a_7	a_6	a_5	a_4	a_3	a_2	a_1
Peso	8	8	4	1	1	1	1
Nodos asignados	2	2	0	1	1	1	1

$$A' = \sum a_i' = 24$$

$$MECGS(A') = 36$$

Fig. 2 Ejemplos de MECGS obtenidos a partir de dos árboles *A* y *A'*

En los ejemplos de la figura 2 puede verse que un árbol *A'* no siempre ofrece una solución para MECGS mejor que otro árbol *A* de mayor peso. Igualmente, en los ejemplos de la figura 3 se observa que, cuando la distribución de los nodos es distinta, dos árboles distintos pueden dar lugar a asignaciones idénticas de los radios de cobertura de los nodos. De ahora en adelante, en las figuras se indicará o bien el nombre de la arista o bien, por simplicidad, su longitud (recuérdese entonces que el peso o coste de la arista es dicha longitud elevada al cuadrado).



n_1	n_2	n_3	n_4
3^2	3^2	3^2	3^2

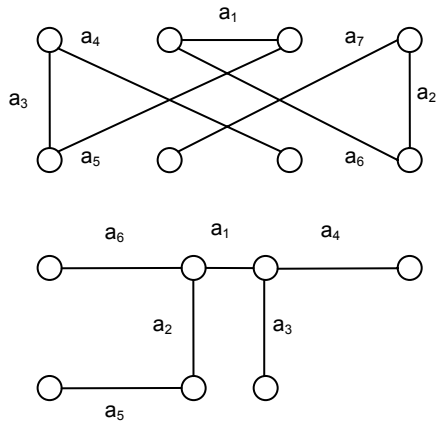
n_1	n_2	n_3	n_4
3^2	3^2	3^2	3^2

n_1	n_2	n_3	n_4
3^2	3^2	3^2	3^2

Fig. 3 Grafos distintos en los que todos los nodos tienen el mismo coste $R_i^2=3^2$

Por otro lado, como las aristas se han ordenado de mayor a menor peso, para pasar de un árbol *A* a la solución del problema MECGS el peor caso lo ofrecen las siguientes secuencias de asignaciones:

$$\text{MECGS(A)} = 2\cdot\left(a_{n-1} + a_{n-2} + \ldots + a_{n/2}\right) \qquad \text{si } n \text{ es par}$$
$$\text{MECGS(A)} = 2\cdot\left(a_{n-1} + a_{n-2} + \ldots + a_{n+1/2}\right) + a_{n-1/2} \qquad \text{si } n \text{ es impar}$$



8 nodos

	a_7	a_6	a_5	a_4	a_3	a_2	a_1
Nodos asignados	2	2	2	2	0	0	0

7 nodos

	a_6	a_5	a_4	a_3	a_2	a_1
Nodos asignados	2	2	2	1	0	0

Fig. 4 Ejemplos de secuencias de asignaciones con el valor más alto posible de ΣR_i^2

En la siguiente proposición se demuestra además que la secuencia de asignaciones no puede ser cualquiera, sino que –además de cumplirse que la arista a_{n-1} ha de tener dos asignaciones, la arista a_{n-2} dos asignaciones o una y el resto de aristas dos, una o cero– dicha secuencia debe cumplir una determinada propiedad.

Proposición 1. Para el problema del MECGS obtenido a partir del MST, no puede obtenerse la siguiente secuencia de asignaciones:

a_{n-1}	a_{n-2}	a_{n-3}	…	…	a_{j+1}	a_j	…	…
2	1	1	1	1	1	0	…	…

Demostración. Si dibujamos las aristas de la secuencia de asignaciones 2,1,1,1,…,1 hasta a_{j+1} formaremos un árbol conexo como el de la figura 5. Si, a continuación, a a_j no le corresponde ninguna asignación, ello significa que sus dos nodos ya tienen asignada una arista (o un radio, como se prefiera) y que por tanto pertenecen ambos al árbol conexo formado por la sucesión inicial. Esto, no obstante, es imposible ya que entonces a_j formará un ciclo con el resto de las aristas y no tendremos inicialmente un árbol. □

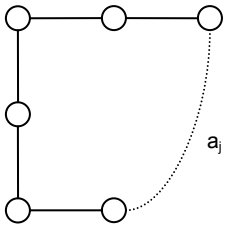


Fig. 5 Formación del grafo a partir de la secuencia de asignaciones 2,1,1,1,…,1

De este modo, para encontrar un 0 en la sucesión, debemos tener una arista posterior a a_{n-1} asignada 2 veces, según se indica en la figura 6. En este caso la nueva arista con doble asignación (dibujada con mayor grosor) ha empezado a formar un nuevo árbol desconectado del anterior y por tanto podemos tener una arista a_j sin asignaciones, incidente a dos nodos pertenecientes a los dos árboles sin conectar (un nodo al primer árbol y el otro nodo al segundo árbol), y que de este modo no forme ningún ciclo.

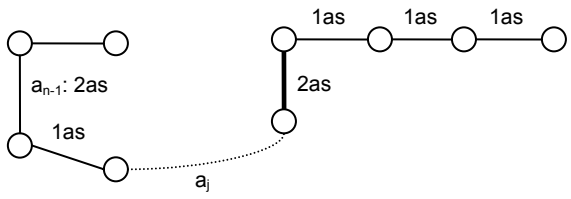


Fig. 6 Una arista distinta de a_{n-1} con dos asignaciones puede permitir una nueva arista a_j sin asignaciones

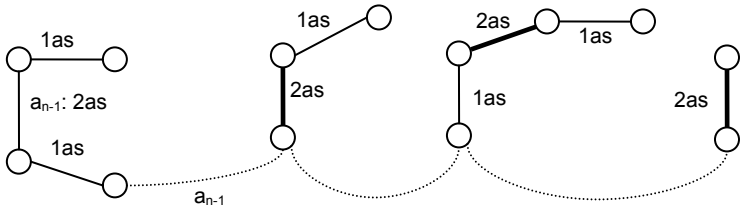


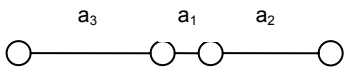
Fig. 7 Cada arista distinta de a_{n-1} con dos asignaciones puede permitir una nueva arista sin asignaciones, señalada en discontinua

De la misma manera, si se tiene una única arista asignada 2 veces (además de la arista de mayor peso del árbol a_{n-1}) y existe otra arista con asignación 0, no puede existir otra arista sin asignación, ya que entonces se estará formando un ciclo. Más en general, a partir de un momento dado podemos tener un número de aristas con asignación 0 igual al número de aristas con asignación 2 (descontando siempre a_{n-1}) menos el número de aristas con asignación 0, según se ilustra en la figura 7. En este caso, tenemos tres aristas (sin contar a_{n-1}) con doble asignación (que en la figura aparecen dibujadas con mayor grosor) que van a permitir por tanto la existencia de tres aristas posteriores (que en la figura aparecen dibujadas en línea discontinua) con asignación igual a cero.

Teorema 1. Dado un grafo *G* con un árbol generador *A* cuyas aristas tienen un peso total de $A=a_{n-1}+a_{n-2}+\ldots+a_1$, la solución para el problema del MECGS obtenida a partir de *A* cumple la siguiente desigualdad:

$$MECGS(A) \geq a_{n-1} + A = a_{n-1} + \sum_{i=0}^{n-1} a_i$$

Demostración. La arista a_{n-1} se asigna siempre a dos nodos, así que en la solución de MECGS aparece dos veces (una para cada nodo). Por otro lado, el resto de aristas se tiene que contar como mínimo una vez, salvo aquellos casos en los que una arista tiene asignación 0. Llamemos a_j a una arista que tiene asignación 0. En este caso, según se ha visto en la Proposición 1, ha de existir una arista de peso mayor asignada dos veces que aporta al sumatorio de MECGS un peso mayor al de a_j. De ahí se obtiene la cota del teorema. □



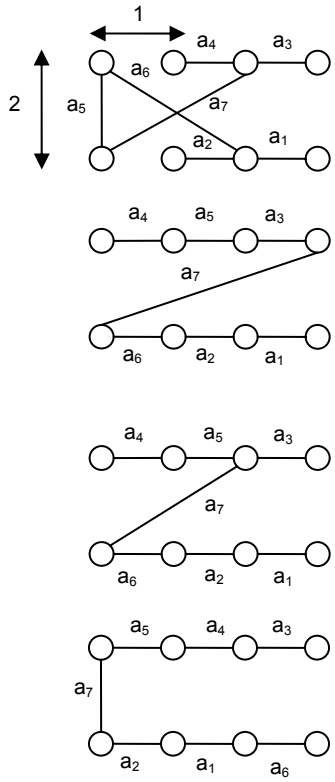
a_3	a_2	a_1
16	9	1
2	2	0

$MST = 26$
$MECGS(MST) = 50$
$MECGS(MST) \geq 16 + 26 = 42$

Fig. 8 Ejemplo de grafo en el que $MECGS(MST) > a_{n-1}+MST$

En el ejemplo de la figura 8 se ve además un caso en que con MST no se cumple la igualdad de la expresión anterior, es decir, en el que $MECGS(A) > a_{n-1}+A$ para *A*=MST. En cambio, en los últimos tres casos de la Figura 9 se cumple la igualdad $MECGS(A) = a_{n-1}+A$ debido a que todos siguen la sucesión de asignaciones 2,1,1,1,…,1.

En el ejemplo de la figura 2 se han visto dos árboles *A* y *A'* generadores de un mismo grafo tales que $\Sigma a_i \geq \Sigma a'_i$ y tales que $a_{n-1} \leq a'_{n-1}$. Es decir, puede darse el caso de un árbol *A* de peso mayor a otro *A'* cuya mayor arista sea sin embargo menor que la de *A'*. En la Proposición 2 se demuestra, sin embargo, que no existe ningún árbol generador cuya arista de peso mayor sea menor a la mayor arista de MST. Este resultado es importante para la demostración de algunas de las proposiciones que se presentan más adelante en este mismo apéndice.



a_7	a_6	a_5	a_4	a_3	a_2	a_1	
8	8	4	1	1	1	1	$A = 24$
2	2	0	1	1	1	1	$MEGS(A) = 36$

a_7	a_6	a_5	a_4	a_3	a_2	a_1	
13	1	1	1	1	1	1	$A = 19$
2	1	1	1	1	1	1	$MECGS(A) = 32$

a_7	a_6	a_5	a_4	a_3	a_2	a_1	
8	1	1	1	1	1	1	$A = 14$
2	1	1	1	1	1	1	$MECGS(A) = 22$

a_7	a_6	a_5	a_4	a_3	a_2	a_1	
4	1	1	1	1	1	1	$A = 10$
2	1	1	1	1	1	1	$MECGS(A) = 14$

Fig. 9 Ejemplos de MECGS obtenidos a partir de árboles

Proposición 2. Dado un grafo *G* y un árbol generador mínimo MST de *G* tal que la arista mayor de MST tiene peso a_{n-1}, la arista de mayor peso de cualquier otro árbol generador de *G* tiene un peso a'_{n-1} mayor o igual a a_{n-1}.

Demostración. Sea *G* un grafo con los pesos de las aristas ordenados de menor a mayor. Supóngase que $a_{n-1} > a'_{n-1}$.

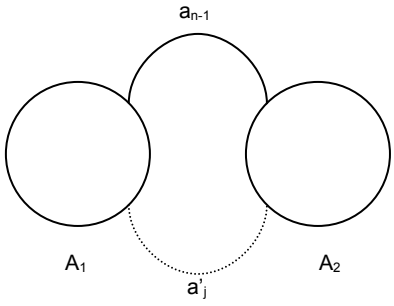


Fig. 10 Componentes conexas A_1 y A_2 de MST unidas por a_{n-1}

A continuación se dibujan las componentes conexas A_1 y A_2 de MST unidas por a_{n-1} (figura 10) y se toma la arista a'_j que en *A'* une estas componentes conexas, i.e. que une un nodo de A_1 con otro de A_2. Por hipótesis tenemos $a'_j \leq a'_{n-1} < a_{n-1}$. De este modo, si se sustituye en MST la arista a_{n-1} por la arista a'_j resulta un árbol generador (ya que no contiene ciclos) de menor peso que MST, lo cual es imposible por definición del MST. De este modo la hipótesis no puede ser cierta y por tanto $a'_{n-1} \geq a_{n-1}$. □

Por otro lado, en la figura 11 se muestra el ejemplo de un grafo para el cual existe un árbol *A'* tal que $a'_{n-1}=a_{n-1}$ pero que tiene mayor peso que MST.

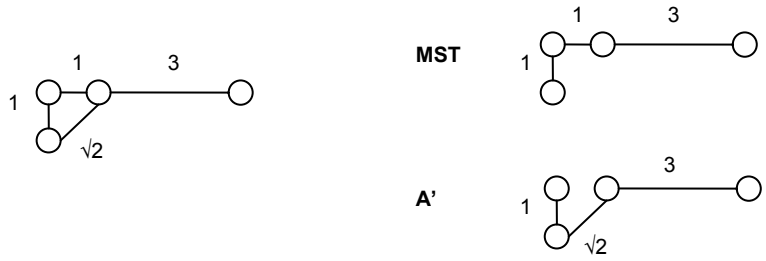
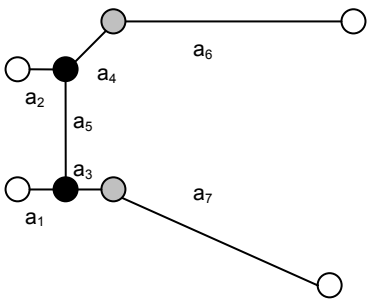


Fig. 11 Ejemplo de árbol A' de mayor peso que MST y tal que $a'_{n-1}=a_{n-1}$

Proposición 3. La solución ofrecida por un árbol *A'* cumple MECGS(A') ≥ a_{n-1}+MST, en donde a_{n-1} es el peso de la mayor arista de MST.

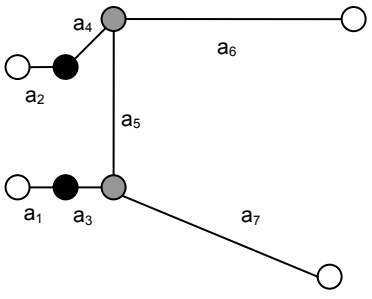
Demostración. Según el Teorema 1 tenemos MECGS(A') ≥ a'_{n-1}+A' y según la Proposición 2 tenemos $a'_{n-1} \geq a_{n-1}$ Como, además, por definición A' ≥ MST, resulta finalmente MECGS(A') ≥ a'_{n-1}+A' ≥ a_{n-1}+MST. □



MST

a_7	a_6	a_5	a_4	a_3	a_2	a_1
25	25	9	2	1	1	1
2	2	2	0	0	1	1

$$MECGS(MST) = 120$$
$$MST = 64$$
$$MST + a_{n-1} = 89$$



A

a_7	a_6	a_5	a_4	a_3	a_2	a_1
25	25	16	2	1	1	1
2	2	0	1	1	1	1

$$MECGS(A) = 105$$
$$A = 71$$
$$MST + a_{n-1} = 96$$
$$MECGS(A) < MECGS(MST)$$

Fig. 12 Ejemplo de grafo en el que MST no da la solución óptima del MECGS

3. MECGS ≠ MST

A pesar de que para algunos de los ejemplos simples vistos hasta ahora el MST ofrece una solución óptima del problema del MECGS, pueden encontrarse grafos en los que esto no es así. De este modo, según los procedimientos de asignación seguidos en el presente apéndice, el MST no siempre ofrece la solución óptima del MECGS.

Teorema 2. MECGS ≠ MST.

Demostración. Por contraejemplo. En la figura 12 se presenta un grafo en el que un árbol *A*, de mayor peso que MST, ofrece una solución mejor que MST para el MECGS. En efecto, al sustituir en MST la arista que une los nodos representados en negro por la arista de mayor longitud que en *A* une los nodos representados en gris, se obtiene un árbol *A* de peso mayor que MST. Sin embargo, de esta forma se ha eximido a los nodos negros de un radio igual al de la arista eliminada y los dos disminuyen su radio de cobertura al pasar de MST a *A*. Por otro lado, aunque los nodos grises tienen una arista nueva, su radio no va a cambiar dado que son incidentes a aristas mayores que la nueva arista. Es por esto que finalmente *A* ofrece una mejor solución para el MECGS que MST, a pesar de tener un peso mayor.□

En suma, lo mismo que sucede para el MECBS, el MST no siempre ofrece una solución óptima del MECGS puesto que no pretende minimizar la suma de todas las aristas, sino la suma de los radios de cobertura de los nodos. En la figura 13 puede verse además un grafo en el que dos árboles generadores mínimos distintos dan lugar a distintas soluciones del MECGS.

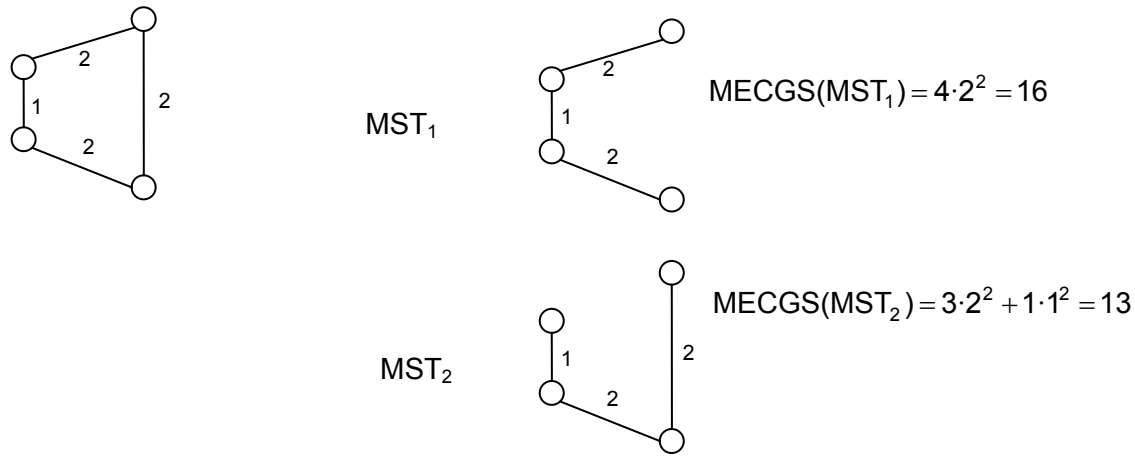


Fig. 13 Ejemplo de grafo en el que dos árboles generadores mínimos dan distintas soluciones del problema del MECGS

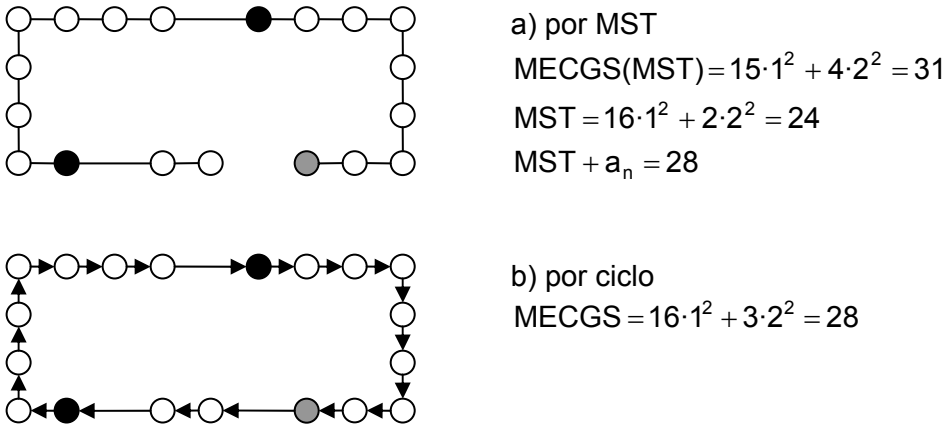
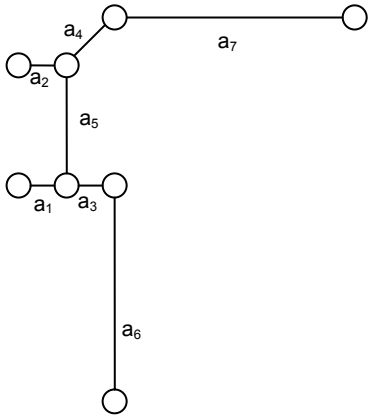


Fig. 14 Ejemplo de grafo en el que MST no da la solución óptima del MECGS y en que la mejor solución la da un ciclo

Por otro lado, para la demostración del Teorema 2 puede presentarse otro contraejemplo en el que la mejor solución se obtiene a partir no de un árbol sino de un ciclo (figura 14). En el ejemplo de la figura resuelto a partir del MST (y siguiendo el procedimiento explicado hasta ahora), los nodos que se encuentran entre dos aristas de radios 1 y 2, tienen un radio de cobertura igual a 2 ya que han de cubrir ambas aristas. Por otro lado, al resolver el problema a partir del ciclo, encontramos una arista más, de modo que el nodo señalado en gris aumenta su radio de cobertura de uno a dos. Pero, por otro lado, al recorrer el grafo en un solo sentido (contrariamente al caso del MST en el que hemos de recorrerlo en los dos sentidos), los nodos representados en negro pasan de tener un radio de cobertura igual a dos a un radio igual a la unidad. Es por esto que finalmente el ciclo ofrece una mejor solución.

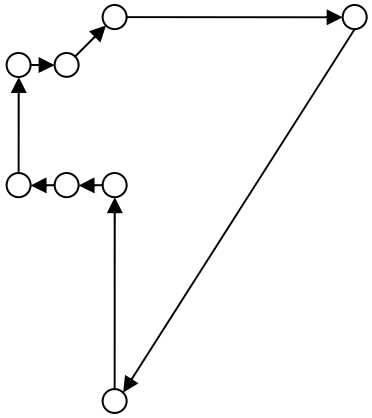


a) por MST

a_7	a_6	a_5	a_4	a_3	a_2	a_1
25	25	9	2	1	1	1
2	2	2	0	0	1	1

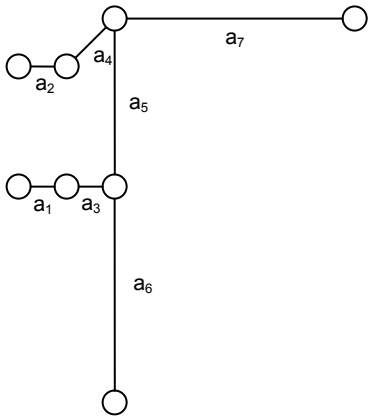
$MST = 64$

$MECGS(MST) = 120$



b)por ciclo

$MECGS = 3 \cdot 1^2 + 1 \cdot 2 + 1 \cdot 3^2 + 2 \cdot 5^2 + 1 \cdot 106 = 170$



c) por A ≠ MST

a_7	a_6	a_5	a_4	a_3	a_2	a_1
25	25	16	2	1	1	1
2	2	0	1	1	1	1

$A = 71$

$MECGS(A) = 105$

Fig. 15 Ejemplo de grafo en que la mejor solución la ofrece un árbol mayor que MST

Dado el ejemplo de la figura 14, cabe preguntarse si el método del ciclo va a dar siempre mejores resultados. Esto no va a ser así en general (aunque existe la excepción de casos muy restrictivos como el del ejemplo anterior) entre otros motivos porque la distribución de los nodos no va a permitir siempre el dibujo sencillo de un ciclo. Más en general, si se pretende dibujar un ciclo a partir del MST u otro árbol cualquiera, sólo va a ser posible cuando el árbol tenga únicamente dos nodos terminales (i.e. dos nodos de grado 1, de tal forma que todos los demás tengan grado 2). En este caso, dichos nodos terminales se unirán con una arista adicional para cerrar el ciclo y recorrerlo en un solo sentido, como en el ejemplo de la figura 14. En otro caso, no siempre va a poder formarse un ciclo o conjunto de ciclos de forma simple. En la figura 15 se presenta un grafo en el que el método basado en el ciclo ofrece una solución peor que la que se basa en los árboles. En este caso, además, puede encontrarse un árbol *A* que ofrece una solución mejor que la de MST. Más en general, en aquellos casos en que dos métodos obtienen la misma solución del MECGS, puede escogerse el método que ofrezca mejores valores para otros parámetros, como por ejemplo el diámetro de la red o la distancia media entre nodos.

4. COTA DE MECGS

A pesar de que en el apartado anterior se ha visto que en determinados casos existen métodos que ofrecen mejores soluciones que el MST (y en el ejemplo de la figura 18 se ve como además dichos métodos pueden llegar a obtener la solución óptima), puede demostrarse, según el siguiente teorema, que ningún método va a ofrecer una solución por debajo de la cota determinada por el MST.

Teorema 3. La solución óptima del problema de subgrafo de gossiping de energía mínima cumple $MECGS|_{ópt} \geq a_{n-1} + MST$, en donde a_{n-1} es el peso de la mayor arista de MST.

Demostración. Si la secuencia de rangos de la solución óptima puede convertirse en un grafo *G* (es decir, si al asignar los radios de cobertura a cada nodo la distribución resultante de aristas es tal que cuando existe a_{ij} también existe a_{ji}) entonces $MECGS|_{ópt} = MECGS(G) \geq MECGS(MST|_G) \geq a_{n-1} + MST$, en donde $MST|_G$ es el árbol generador mínimo calculado sobre el grafo *G* y MST es el árbol generador mínimo calculado sobre el grafo original completo (en el que todos los nodos ven a todos los nodos, según el procedimiento seguido en el presente proyecto).

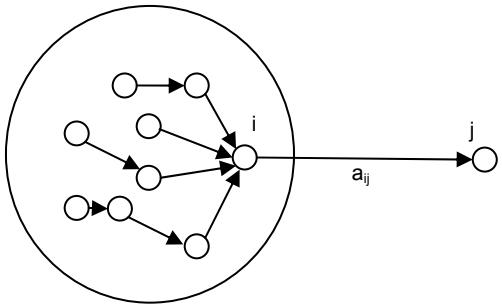


Fig. 16 Separación de la arista a_{ij} de mayor peso

Por otro lado, si la secuencia de rangos de $MECGS|_{ópt}$ no da como resultado un grafo, el teorema puede demostrarse mediante otro procedimiento que también es generalizable para el caso anterior. Se dibuja en primer lugar, como indica la figura 16,

el grafo resultante de aplicar los radios de cobertura de la solución óptima y a continuación se aísla la arista a_{ij} de mayor peso que, como se ve más adelante, ha de ser mayor o igual a la arista a_{n-1} de mayor peso de MST. La arista a_{ij} va del nodo *i* al nodo *j*.

A continuación, para cada nodo distinto de *i* se dibuja su camino más corto hasta *i*. El resultado es, por construcción, un árbol *A*, ya que si un nodo tuviera más de un camino hasta *i* se podrían eliminar aristas hasta dejar un único camino de longitud mínima, lo que da como resultado final un árbol. Al finalizar esta operación, cada nodo tiene asociada una arista de salida menor o igual al radio inicial asignado por $MECGS|_{ópt}$ (puesto que todas las aristas vienen de un grafo dibujado según los rangos de $MECGS|_{ópt}$ del que además se han eliminado algunas aristas). De este modo, todas las aristas de *A* (en las que no hay que incluir la arista de mayor peso a_{ij}) forman un árbol generador, de tal forma que la suma de sus pesos (recogida en el sumatorio de la siguiente expresión) ha de ser mayor o igual a la de un árbol generado mínimo, lo que da:

$$MECGS|_{ópt} \geq a_{ij} + \sum_{k \neq i} a_k \geq a_{ij} + MST$$

Por otro lado, como *A* es un árbol generador ha de tener como mínimo, según la Proposición 2, una arista de peso mayor o igual a a_{n-1}, y como además la arista a_{ij} es la de mayor peso del grafo original y por tanto ha de ser mayor que cualquier arista del árbol *A*, tenemos que $a_{ij} \geq a_{n-1}$ y por tanto:

$$MECGS|_{ópt} \geq a_{ij} + MST \geq a_{n-1} + MST. \square$$

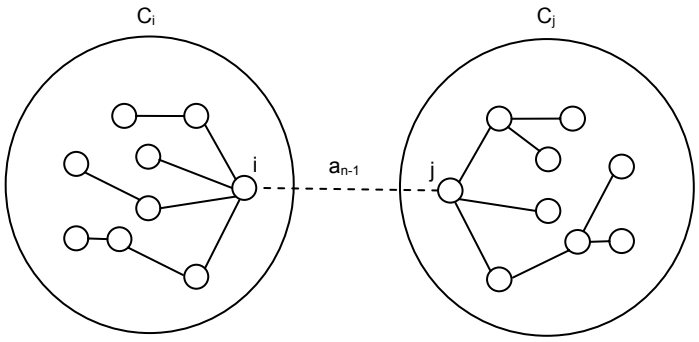
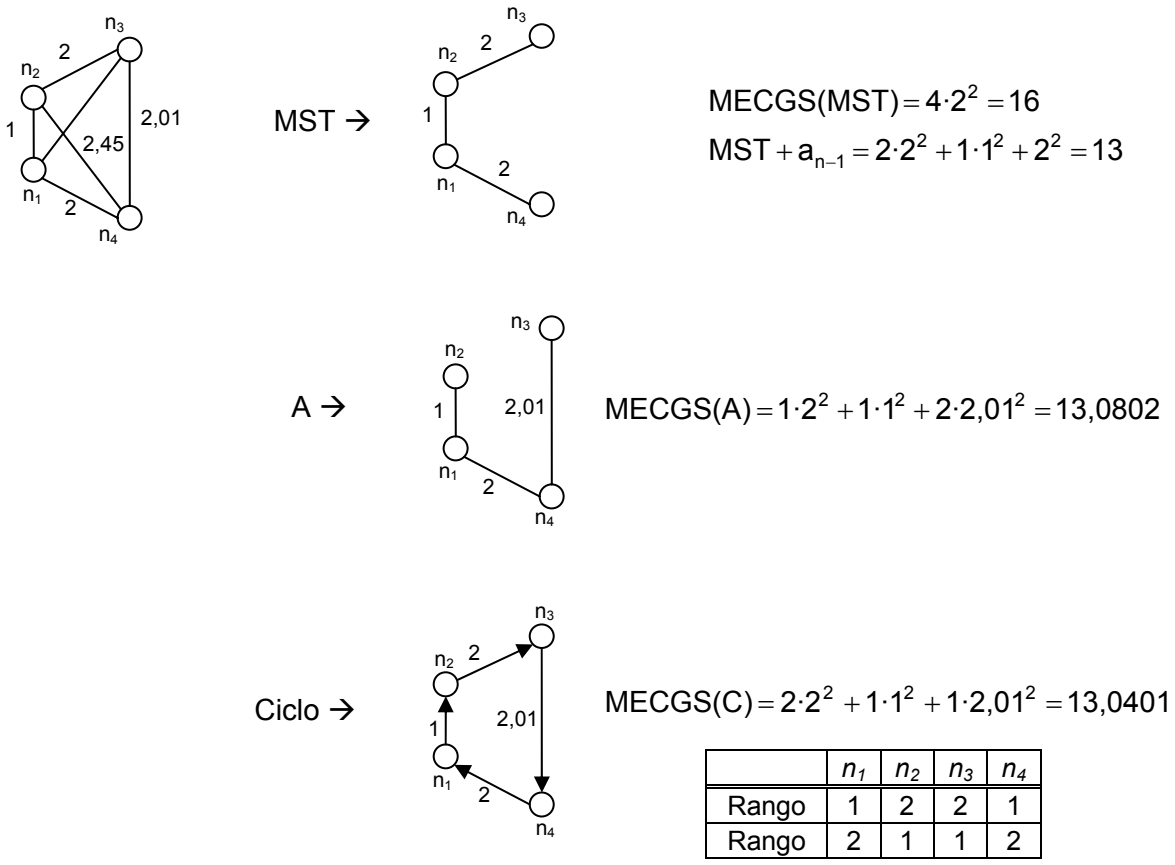


Fig. 17 Componentes conexas C_i y C_j de la solución óptima

De la anterior demostración se deduce que en $MECGS|_{ópt}$ siempre existen dos asignaciones de radios, i.e. dos aristas, de valor mayor o igual a a_{n-1} (esas aristas, con todo, pueden ser simétricas, por ejemplo a_{ij} y a_{ji}). Esto es así ya que si en el MST se separan los nodos *i* y *j* unidos por la arista a_{n-1} de mayor peso y se separan asimismo sus componentes conexas C_i y C_j, entonces en la solución de $MECGS|_{ópt}$ tendremos que ir de C_i a C_j, y para ello necesitaremos una arista (y por tanto una asignación) mayor o igual que a_{n-1} ya que un nodo *i'* de C_i no puede estar a distancia de un nodo *j'* de C_j menor a a_{n-1} (en este caso podría sustituirse a_{n-1} por $a_{i'j'}$ y formarse un árbol de peso menor que el MST). De forma análoga, para ir desde C_j a C_i se necesita una asignación mayor que a_{n-1}, según se indica en la figura 17. Como las asignaciones del resto de los nodos tampoco pueden ser inferiores a las del MST (puesto que en otro caso podríamos formar con ellas un árbol generador de peso menor a MST), entonces:

$$MECGS|_{ópt} \geq a_{n-1} + MST$$

La figura 18 muestra, además, un grafo en el cual no se cumple la igualdad del Teorema 3, es decir, en el cual $MECGS|_{ópt} > a_{n-1}+MST$. En este caso vemos que existe un árbol *A* que ofrece una solución mejor que MST y que puede obtenerse una solución todavía mejor a partir de un ciclo. En los párrafos siguientes se demuestra que dicha solución obtenida mediante un ciclo, que es mayor que $a_{n-1}+$ MST, es óptima, de modo que finalmente tenemos $MECGS|_{ópt} > a_{n-1}+$ MST. Para simplificar el análisis llamamos a los nodos n_1, n_2, n_3 y n_4.



	n_1	n_2	n_3	n_4
Rango	1	2	2	1
Rango	2	1	1	2

Fig. 18 Ejemplo de grafo para el cual $MECGS > a_{n-1}+MST$

Al final de la figura 18 se muestra la tabla de rangos de la tercera solución, obtenida a partir del ciclo, en la que el primer nodo tiene rango uno puesto que sólo alcanza su nodo más próximo, el segundo tiene rango dos porque su radio de cobertura alcanza dos nodos (n_1 y n_3, aunque en la figura por simplicidad sólo se haya dibujado la arista hasta n_3), y así sucesivamente. Igualmente, se ha añadido la tabla de rangos que se obtiene al recorrer el ciclo en sentido contrario y que ofrece una solución equivalente, dada la simetría con que hemos distribuido los puntos del grafo.

Si a continuación tratamos de obtener una solución mejor que la que ofrece el ciclo, entonces uno de los nodos (n_2 y n_3) que tiene rango 2 ha de pasar a tener rango 1, para disminuir la función de coste final. A continuación analizamos todas las posibilidades en que uno de dichos nodos (o los dos) tiene rango uno, para comprobar que o bien no ofrecen conectividad o bien tienen una función de coste mayor a la del ciclo, que como se indica en la figura 18 es igual a 13,0401.

En primer lugar, se analizan los casos en que n_2 tiene rango 1. En este caso, n_1 ha de tener como mínimo rango 2 ya que si no los nodos n_1 y n_2 quedan aislados del resto de la red y se pierde la conectividad. Empezamos analizando, entonces, los casos en los que n_1 tiene rango 2.

n_1	n_2	n_3	n_4	Conectividad	Coste
2	1	1	1	No	
2	1	1	2	Si	13,0401 (ciclo recorrido en sentido inverso)
2	1	1	3	Si	> coste(2112)=13,0401
2	1	2	1	No	
2	1	2	2	Si	> coste(2112)=13,0401
2	1	2	3	Si	> coste(2112)=13,0401
2	1	3	1	No	
2	1	3	2	Si	> coste(2112)=13,0401
2	1	3	3	Si	> coste(2112)=13,0401

En las tres secuencias de rangos 21X1 no hay conectividad con independencia del rango de n_3 ya que, según se observa en la figura 19, n_3 siempre permanece aislado, i.e. desde cualquiera de los otros tres nodos no puede alcanzarse n_3.

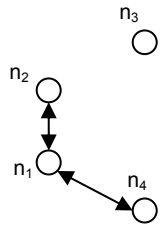


Fig. 19 Falta de conectividad de n_3 en 21X1

Por otro lado, el coste de la secuencia de rangos 2113 ha de ser necesariamente mayor que el coste de la secuencia 2112 puesto que los tres primeros nodos tienen el mismo rango y el cuarto nodo tiene rango mayor. Del mismo modo, las secuencias 2122, 2123, 2132 y 2133 tienen un coste mayor que el de 2122. Este mismo argumento se ha aplicado en las tablas siguientes. A continuación se presenta la tabla en la que n_2 mantiene el rango 1 y n_1 tiene rango 3

n_1	n_2	n_3	n_4	Conectividad	Coste
3	1	1	1	Si	$1 \cdot 2{,}45^2 + 2 \cdot 2^2 + 1 \cdot 1^2 > 13{,}0401$
3	1	1	2	Si	
3	1	1	3	Si	
3	1	2	1	Si	
3	1	2	2	Si	coste > coste(3111) > 13,0401
3	1	2	3	Si	
3	1	3	1	Si	
3	1	3	2	Si	
3	1	3	3	Si	

De este modo, si se fija el rango de n_2 a uno, todas las posibilidades o bien no tienen conectividad, o superan el coste mínimo de la secuencia 1221 que, como hemos dicho, es de 13,0401. En rigor, se ha encontrado una secuencia 2112 que da el mismo valor pero que, como se ha indicado, es una solución equivalente a la de 1221 puesto que recorre el mismo ciclo en sentido contrario. A continuación se examinan los casos en los que el rango de n_3 vale 1. Por simplicidad se examinan tres tablas según n_1 tenga rango 1, 2 o 3.

n_1	n_2	n_3	n_4	Conectividad	Coste
1	1	1	1	No	En los tres casos rango1=rango2=1
1	1	1	2	No	
1	1	1	3	No	
1	2	1	1	No	
1	2	1	2	No	
1	2	1	3	No	
1	3	1	1	Si	$1 \cdot 2{,}45^2 + 2 \cdot 2^2 + 1 \cdot 1^2 > 13{,}0401$
1	3	1	2	Si	> coste(1311) > 13,0401
1	3	1	3	Si	> coste(1311) > 13,0401

En los tres primeros casos de la anterior tabla no hay conectividad ya que los nodos n_1 y n_2 tienen rango 1 y sólo pueden comunicarse entre ellos. Por otro lado, de forma simétrica a lo que sucedía con las secuencias 21X1 de la figura 19, en las tres secuencias de rangos 121X no hay conectividad puesto que n_4 permanece aislado (figura 20).

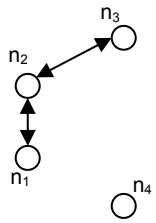


Fig. 20 Falta de conectividad de n_4 en 121X

n_1	n_2	n_3	n_4	Conectividad	Coste
2	1	1	1	analizados en la primera tabla	
2	1	1	2		
2	1	1	3		
2	2	1	1	Si	$4 \cdot 2^2 > 13{,}0401$
2	2	1	2	Si	coste > coste(2211) > 13,0401
2	2	1	3	Si	
2	3	1	1	Si	
2	3	1	2	Si	
2	3	1	3	Si	

n_1	n_2	n_3	n_4	Conectividad	Coste
3	1	1	1	} analizados en la segunda tabla	
3	1	1	2		
3	1	1	3		
3	2	1	1	Si	} coste > coste(3111) > 13,0401
3	2	1	2	Si	
3	2	1	3	Si	
3	3	1	1	Si	
3	3	1	2	Si	
3	3	1	3	Si	

De este modo, todas las combinaciones posibles o bien no tienen conectividad o bien tienen una función de coste mayor que la que ofrecen las secuencias de rangos 1221 y 2112 (que como se ha dicho son equivalentes puesto que 2112 recorre el mismo ciclo que 1221, pero en sentido contrario), de modo que la solución de dichas secuencias es óptima y por tanto el coste de la solución óptima es $13,0401 > MST + a_{n-1}$. Este resultado ha sido de especial importancia en el presente proyecto, puesto que aunque en las pruebas realizadas se han comparado las soluciones obtenidas con la cota anterior, se ha tenido en cuenta que dicha cota no ha de corresponderse necesariamente con el valor de la solución óptima del problema.

5. CONCLUSIONES

En el presente apéndice hemos visto que, en el mejor de los casos, la solución del MECGS obtenida a partir del árbol generador mínimo MST da un valor $MECGS(MST) = a_{n-1} + MST$ y que por tanto $MECGS(MST) \geq a_{n-1} + MST$. También se ha visto que existen grafos para los cuales otros árboles o digrafos pueden dar mejores soluciones que MST, y por tanto $MECGS \neq MST$. Aún así, no existen soluciones con un valor por debajo de la cota $a_{n-1} + MST$ determinada por el árbol generador mínimo, por lo que $MECGS|_{ópt} \geq a_{n-1} + MST$.

De todo esto se deduce que si un método encuentra una solución de valor $a_{n-1} + MST$, dicha distribución de radios ha de ser óptima puesto que, como se ha explicado, no existen soluciones por debajo de dicho umbral.

Igualmente, en aquellos grafos para los cuales existen métodos que dan mejores soluciones que MST, se tiene que $MECGS (MST) > MECGS|_{ópt} \geq a_{n-1} + MST$, es decir, $MECGS (MST) > a_{n-1} + MST$, en cuyo caso la sucesión de asignaciones para MST no puede ser 2,1,1,1,…,1. Por último, se ha visto que existen grafos para los cuales la solución óptima del MECGS es mayor que la cota calculada, es decir, para los que se cumple $MECGS|_{ópt} > a_{n-1} + MST$.