Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Localización de antenas wifi y de telefonía móvil

Autor: Kevin Hernández Díaz

Tutor: Juan Antonio Mesa López-Colmenar

Dep. Matemática Aplicada II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Localización de antenas wifi y de telefonía móvil

Autor:
Kevin Hernández Díaz

Tutor:
Juan Antonio Mesa López-Colmenar
Catedrático de Universidad

Dep. Matemática Aplicada II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Trabajo Fin de Grado:

Localización de antenas wifi y de telefonía móvil

Autor: Kevin Hernández Díaz

Tutor: Juan Antonio Mesa López-Colmenar

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis amigos

"Any sufficiently advanced technology
is indistinguishable from magic."

- Arthur C. Clarke -

Resumen

En este trabajo se considera el problema de localizar tanto estaciones bases de telefonía móvil como wifi para proporcionar cobertura a un sistema con una estructura de tipo red. El sistema mallado se representa sobre un plano en el cual se conoce la ubicación de un conjunto de coordenadas que representan los puntos de inicio y final de caminos o carreteras. Estos puntos se codifican mediante dos matrices origen-destino que representan las coordenadas en los ejes cartesianos del plano y generan una red de segmentos en los que existe una demanda de servicios telefónicos. A partir de esta demanda de servicios el operador le asigna una prioridad dentro del sistema.

El objetivo del trabajo es la implementación en MATLAB de un algoritmo heurístico que localice un conjunto de estaciones base sobre el plano para satisfacer la demanda de servicios. Para ello, se genera un conjunto de puntos de forma simétrica que cubra el área del plano, puntos que servirán de lugares potenciales para la instalación de antenas. El algoritmo consiste en fijar las antenas a colocar, de forma secuencial buscando maximizar la cobertura proporcionada por las antenas.

Este algoritmo tiene en cuenta el solapamiento entre las áreas de cobertura de las antenas y proporciona a la operadora flexibilidad al respecto. Permite decidir si se incentiva o penaliza el solapamiento mediante la asignación de pesos a los diferentes tipos de cobertura, diferenciando entre cobertura simple, doble o triple, que corresponden a la cobertura proporcionada por una, dos y tres o más antenas respectivamente.

Este modelo no tiene en cuenta el entorno, puesto que el alcance o cobertura que proporciona una antena depende de los obstáculos que estén entre el teléfono y la antena, la altura de esta, las interferencias, reflexiones...etc. Además de la posibilidad de que no se pueda instalar una antena en el lugar donde nos indique el programa debido a licencias, permiso de los propietarios del terreno o que la compensación económica al colocar una antena en ese lugar, bien a los dueños del terreno o a quienes se vean afectados por la instalación sea demasiado elevada. Por ello, aunque este algoritmo proporciona una buena primera aproximación al problema, será necesario un estudio posterior sobre la viabilidad de la instalación de las antenas en los lugares propuestos.

Abstract

This paper considers the problem of locating both mobile base stations and Wi-Fi routers to provide coverage to a system with a network-like structure. The system is represented on a plane in which the location of a set of coordinates that represent the points of start and end of roads or highways are known. These points are encoded by two origin-destination matrices that represent Cartesian coordinates in the plane and generate a network of segments in which there is a demand for telephone services. Then the operator assigns a priority to each segment based on their demands.

The aim of this work is the implementation in MATLAB of a heuristic algorithm to locate a set of base stations on the plane to supply the demand for services. To do this, a set of points are located covering the total area of the plane, this points serve as potential sites for the installation of antennas. The algorithm fix the antennas to be placed and then sequentially seeks to maximize the coverage provided for each one.

This algorithm takes into account the overlap between the coverage areas of the antennas and provides the operator flexibility about it. It allows decide whether incentives or penalizes overlap by assigning weights to the different types of coverage, differentiating between single coverage, double or triple, corresponding to the coverage provided by one, two and three or more antennas respectively.

This model does not considers environment issues, since the scope or coverage provided by an antenna depends on the obstacles that are between the phone and the antenna, the height of the antenna, interference, reflections ... etc. In addition to the possibility that it may can not be installed on where the program tell us to because of licensing, permission from the owners of the land or because the financial compensation to those affected by the installation is too high. Therefore, although this algorithm provides a good first solution to the problem, further study on the feasibility of installing the antennas in the proposed locations will be necessary.

Índice

Índice de Figuras

1 INTRODUCCIÓN

We are all now connected by the Internet, like neurons in a giant brain.

- Stephen Hawking -

La aparición de los *Smartphone* ha revolucionado el uso de la telefonía móvil en los últimos años. En 2014 la penetración de este servicio alcanzaba las 95,5 líneas por cada 100 habitantes, por lo que existían casi tantas líneas móviles como habitantes del planeta. El crecimiento de la telefonía móvil ha estado liderado por los países en vías de desarrollo, países en los que la penetración ha crecido 21,7 puntos porcentuales entre 2010 y 2014 [1].

Sin embargo, el aspecto más notable de esta revolución se encuentra en el uso del internet móvil, siendo el servicio de comunicaciones que más ha crecido entre 2010 y 2014, con una tasa de crecimiento anual compuesto del 29%. Europa, América y la región CIS son las que alcanzan mayores niveles de penetración de la banda ancha móvil [1].

Solamente en España, a nivel de equipamiento individual, el *Smartphone* es el dispositivo electrónico que mayor penetración alcanza (59,3%). Siendo el porcentaje de hogares que disponen de telefonía móvil el primer trimestre de 2014 del 95,3% y entre la población de 15 años y más este porcentaje alcanza el 87,8%. La conexión a Internet a través del teléfono móvil es utilizada por ocho de cada diez hogares mientras que el acceso a través de *tablets* alcanza al 41% de los hogares [1]. En la figura 1-2 se muestra una previsión del tráfico de datos a través de las redes móviles desde 2015 al 220 [17].

Este aumento del uso de internet móvil ha ido de la mano del uso de las redes sociales, según "El informe de La sociedad en red" [1]. Participar en redes sociales es uno de los principales usos de Internet de los europeos, ya que el 46% declara acceder a la Red con este fin. En España el 51% de los usuarios de Internet declaran participar en este tipo de servicios. Este desarrollo conjunto ha permitido que las personas estén conectadas en todo momento mediante el uso de diferentes aplicaciones de los *Smartphones*. Algunas de las principales redes sociales como son Facebook, Twitter o Instagram son superadas en uso por dispositivos móviles frente al ordenador [2]. Tal como muestra la figura 1-1.

Sin embargo, no sólo los usuarios individuales se han aprovechado de la situación, en 2014 el 36% de las empresas europeas utilizaban estos servicios, 6 puntos porcentuales más que en 2013. En 2014 el porcentaje de

las empresas que facilitaba dispositivos portátiles al menos al 20% de sus empleados se situó en el 25% [1].

Además, tal aumento de la demanda de los servicios ha contribuido a la aparición de nuevas operadoras, que junto con el desarrollo tecnológico ha permitido el abaratamiento de los servicios de cara al usuario, lo que ha contribuido a su expansión.

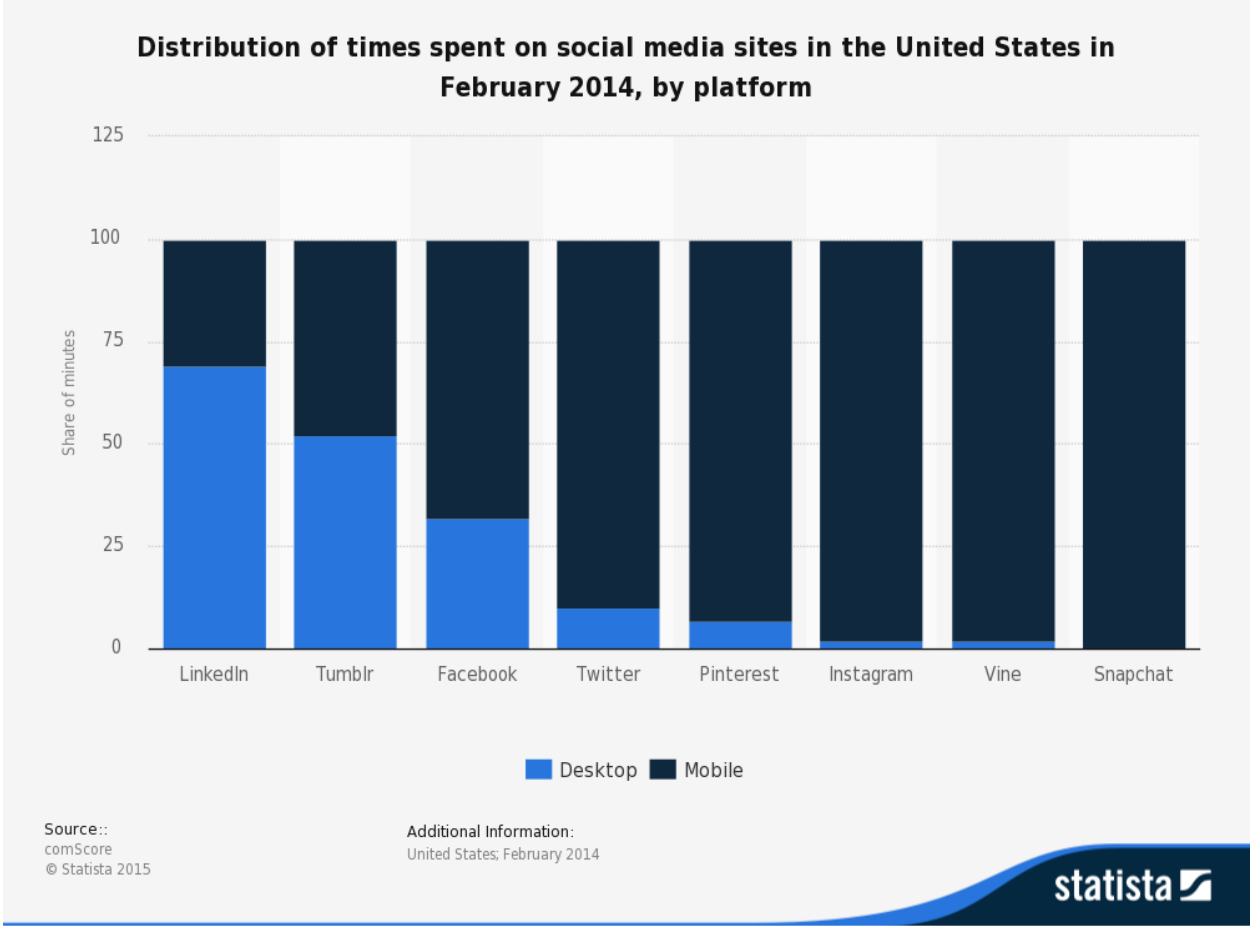

Figura 1-1

El uso de los *Smartphone* ha causado la aparición del negocio de las aplicaciones móviles, en 2015 Apple publicó que los ingresos derivados de las compras de *apps* de su *market* en 2015 fue de 10 billones de dólares [3] y, según estudios, para 2017 se espera que las ganancias de las aplicaciones móviles a nivel mundial supere los 77 billones de dólares. [4]

Solo en el *Google Play Store* en Febrero de 2016 había disponibles cerca de 2 millones de aplicaciones, el doble que en 2013 [5]. Este aumento se debe a la oportunidad de mercado y a la competencia por parte de los desarrolladores de aplicaciones que se esfuerzan por sacar aplicaciones más completas y con mejores características que los demás.

Este desarrollo masivo de aplicaciones es el principal objetivo y uso tanto de los *Smartphones* como de la red móvil. En julio de 2016 'Pokemon Go' una aplicación lanzada ese mismo mes se convirtió en una de las palabras más buscadas de la red por el buscador google [6], convirtiéndose en un fenómeno mundial que saturó los servidores de la compañía y, por consiguiente, produjo un aumento importante en el uso de la red móvil.

Este desarrollo ha llevado a las operadoras a fortalecer sus infraestructuras para aguantar la demanda creciente de los usuarios, tanto en los entornos urbanos como en las carreteras interurbanas ya que el teléfono móvil se ha convertido en el principal método de distracción para los viajes largos, tanto para los pasajeros haciendo uso de las redes sociales o juegos, como de los conductores que lo utilizan tanto para GPS, aplicaciones de música o podcasts, que van sustituyendo a la radio.

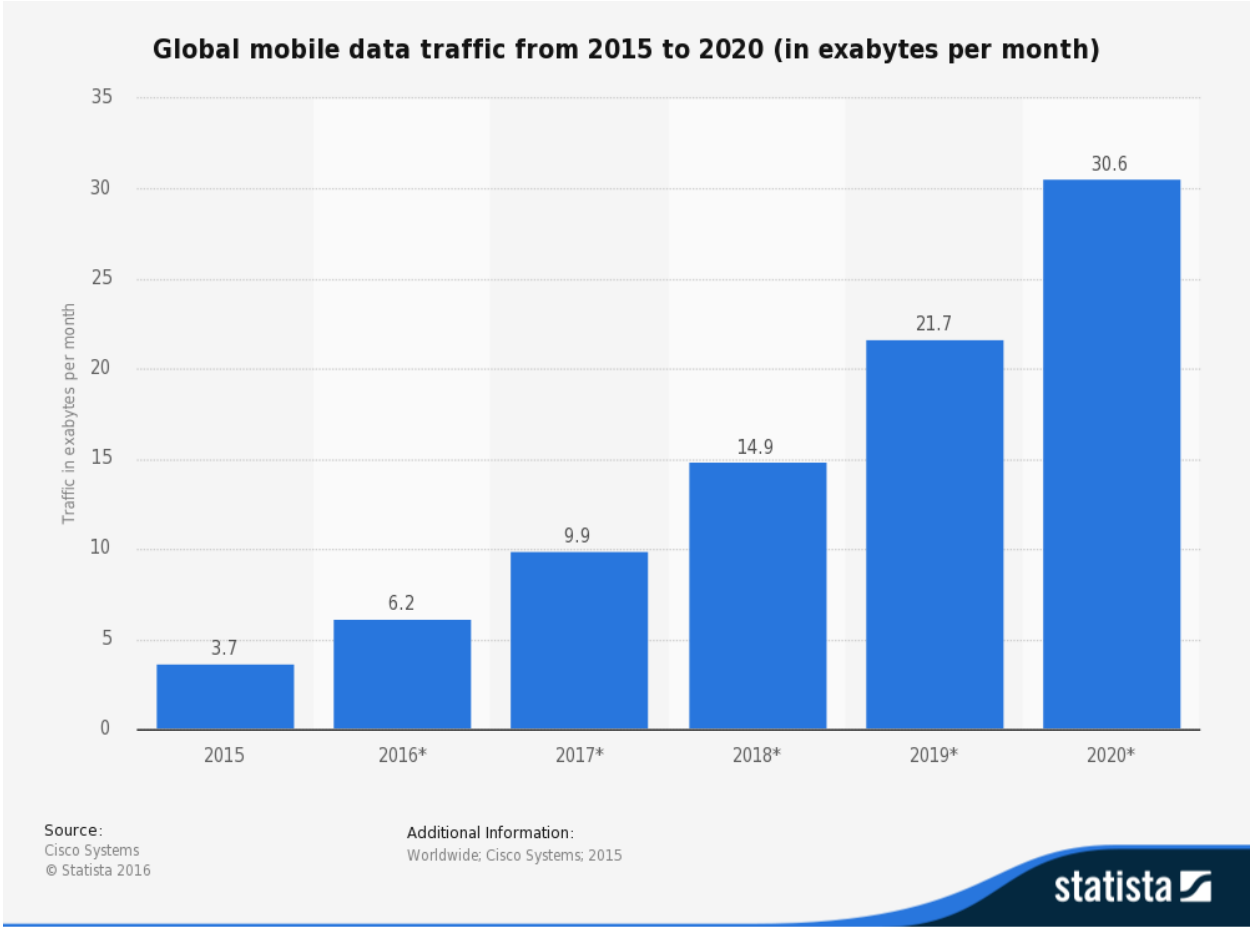

Figura 1-2

Como vemos en la figura 1-2, el tráfico de datos previsto para los próximos años crece de forma exponencial, esto supone el desarrollo y despliegue de una tecnología por parte de las operadoras capaz de facilitar a los usuarios esas capacidades.

En 2013 empezó el despliegue de la red 4g en España por parte de las principales operadoras con el objetivo de proporcionar unas velocidades que satisficieran las demandas y a su vez una mejora en la eficiencia en el uso de las bandas de frecuencia que les corresponde, permitiendo así una mejora tanto para los usuarios como para las empresas.

Sin embargo, no fue hasta la liberación de la banda de 800 MHz por parte de la industria televisiva que las operadoras pudieron aprovechar el potencial de las redes 4g, que se había visto limitada en su expansión a los núcleos urbanos más importantes. Así comenzó el despliegue de la red 4g+ que permitía a las operadoras extender el radio de cobertura al usar la banda de los 800Mhz, que al tener una menor frecuencia de trabajo permitía un mayor alcance y mejor cobertura en interiores que con las bandas de 1.800 MHz o 2.600 MHz utilizadas antes. Además de un mayor ancho de banda al tener otra banda de frecuencia añadida [7].

Ya se está investigando sobre la siguiente generación de telefonía, las redes 5g. Sin embargo, no se espera que esta nueva red se empiece a desplegar hasta el 2020 [8]. Mientras tanto las operadoras tienen otras opciones para mejorar los servicios ya existentes, una de estas soluciones son las llamadas micro células, que consiste en la instalación de antenas de corto alcance para dar mayor cobertura a una determinada zona.

Este tipo de tecnología serviría para dar mayor potencia a zonas puntuales con una mayor demanda como son las playas en verano o por ejemplo los estadios. Esta tecnología a pesar de tener un coste material bajo, se dificulta debido a que los costes de permisos de instalación en tejados y otras zonas son muy caros.

De aquí se deriva la importancia de este trabajo, intentar optimizar el rango de cobertura proporcionado por una antena en una región reduciría la necesidad de instalación de este tipo de antenas temporales así como proporcionar una mayor velocidad a los usuarios. De eso se deriva un ahorro monetario por parte de la operadora, así como una mejora en la imagen de cara al público al proporcionar un mejor servicio a sus usuarios en cuanto a cobertura y velocidad. Además, si nuestra red es suficientemente potente podemos vender parte de nuestra capacidad a otra operadora menor. Por lo tanto el análisis de la localización de las antenas de telefonía móvil es la piedra angular sobre la que recae la operadora.

2 ANTECEDENTES

Getting information off the Internet is like taking a drink from a fire hydrant.

- Mitchell Kapor -

A continuación se expondrán otros trabajos relacionados con el análisis de cobertura de un sistema. No se pretende una revisión exhaustiva del asunto, sino mostrar brevemente otras técnicas estudiadas para la aproximación a este tipo de problemas, tanto de telefonía móvil como con otro fin.

Uno de los pilares fundamentales de este tipo de artículos es el modelo desarrollado por Church y ReVelle [9], Maximal Covering Location Problem (MCLP), en especial los algoritmos GA y GAS. Este modelo ha probado ser uno de los más útiles tanto teóricamente como desde un punto de vista práctico.

- *Greedy Adding (GA) Algorithm*: este algoritmo parte de una solución vacía y va colocando instalaciones una a una en aquel sitio que cubre la mayor cantidad de población que no cubra la anterior instalación, continuando este proceso hasta que todas las instalaciones se hayan colocado o se cubra toda la población. Este algoritmo es óptimo en el caso de una única instalación, sin embargo para mas instalaciones no es seguro que sea la solución óptima, ya que una instalación colocada al principio del proceso puede no estar justificada más adelante.
- *Greedy Adding with Substitution (GAS):* este algoritmo usa el mismo proceso que el GA, pero en cada interacción comprueba que cada una de las instalaciones ya colocadas no mejore la solución al cambiarla a otro sitio libre, si este es el caso cambia dicha instalación al sitio con la mayor mejora del objetivo y continúa el proceso.
- *Linear Programming*: es posible usar la programación lineal para MCLP, sólo hay que restringir que las variables no puedan ser negativas. Siendo los resultados posibles de las variables: Todas ceros o unos, en cuyo caso se ha alcanzado el óptimo, o que algunas variables sean racionales decimales o mayor que 1, en cuyo caso se deberá aplicar distintos métodos para hallar una solución del tipo cero o uno, por ejemplo mediante Branch & Bound.

En el artículo sobre el cual se ha basado este trabajo [10], se propone otra aproximación al problema de la localización de estaciones base. En este método se considera que la demanda está sobre nodos que representan la demanda de una zona o sobre los arcos que representan una demanda en movimiento. Para la solución de este problema los autores desarrollaron un nuevo algoritmo (*GPAH - Greedy paired adding heuristic*) basado en MCLP. En este algoritmo se busca localizar un par de estaciones base dentro de los sitios candidatos para maximizar la función objetivo considerando que un arco cubierto parcialmente no mejora el resultado.

Un trabajo anterior se realizó en el articulo M.R. Akella et al (2005) [11] donde se proponen 5 posibles técnicas para solucionar el problema de localización de estaciones base en una red celular con necesidades de cobertura para emergencias. Estas 5 técnicas son las siguientes:

- *Deterministic Addition (DA) heuristic*: solución basada en el algoritmo MCLP de Church and ReVelle (1974) [9]. Con este método, primero se van colocando las BS en donde se maximice la cobertura de los nodos de emergencia, considerando que al cubrir bien un nodo es posible cubrir parcialmente otro, y en cada paso se quitan tanto el nodo como la demanda cubierta por la BS colocada, siguiendo este proceso hasta que todos los nodos de emergencia estén cubiertos. Una vez que todos éstos estén cubiertos se usa el método MCLP para maximizar la cobertura de los nodos de demanda.
- *Probabilistic Addition (PA) heuristic*: en este método se le asigna una probabilidad a seleccionar un determinado nodo de emergencia para que sea cubierto por la actual BS a colocar, siendo esta probabilidad inversamente proporcional al número de estaciones base cubriendo ese nodo de emergencia. Este análisis heurístico continúa hasta encontrar una solución viable o alcanza un número de interacciones máximo. La principal ventaja de este heurístico es que, de existir una solución viable, es muy probable que devuelva una solución.
- *Set Max Cover (SMC) deterministic heuristic*: es una ligera modificación del DA que se divide en dos fases, la primera se concentra totalmente en cubrir los problemas de cobertura de los nodos de emergencia, mientras que la segunda fase busca maximizar la cobertura de los nodos de demanda
- *Set Max Cover (SMC) probabilistic heuristic*: modificación del SMC, este método intentará evitar quedarse atrapado en un óptimo local mientras realiza la búsqueda.
- *Lagrangean heuristic*: en este método se usa la relajación lagrangiana y el método del subgradiente para hallar una solución al problema.

Un artículo especialmente importante en la localización de estaciones base de telefonía móvil es el realizado por Berman, O.; Krass, D.; Drezner, Z (2003) [12]. En este artículo se trata el problema del radio de cobertura de las instalaciones en este tipo de modelos, ya que si por ejemplo, el radio de cobertura de un supermercado en una población se toma como 5Km se considera que una persona que está a 4.9 Km entra dentro del rango de esa instalación, mientras que una persona que se encuentre a 5.1Km no se consideraría. Este tipo de problemas es especialmente importante en las antenas de telecomunicaciones ya que el alcance de ésta depende del entorno (obstáculos, reflexiones, interferencias...), de esta forma puede ocurrir que una persona dentro del radio establecido en el modelo no tenga servicios (o a una calidad por debajo de la calidad de servicio establecida por nosotros), mientras que en otro sitio una persona que se encuentre más alejada del radio de alcance de la antena establecido por nosotros si tenga y no se incluya en el modelo.

En relación con el artículo anterior, Karasakal, O y Karasakal, E (2004) [13] tratan el asunto del umbral crítico utilizado en MCLP en el cual el valor de la cobertura cambia de cubierto a no cubierto a una distancia específica. En cambio propone una solución basado en relajación lagrangiana, en la cual introduce un rango de distancia donde se permite el cambio de cubierto a no cubierto, a este tipo de cobertura la llama cobertura parcial.

Otro artículo interesante para este trabajo es el realizado por Murray, Alan; Wei, Ran; Batta, Rajan (2014) [14], el problema tratado en este artículo persigue minimizar el número de instalaciones a localizar para satisfacer la demanda. Este modelo se basa en el clásico LSCP. Se proponen dos modelos:

- *Arc covering problem (ACP):* en este modelo el objetivo es minimizar el número de instalaciones a colocar. Las demandas son representadas en arcos, al contrario que LSCP y cada arco tiene que estar cubierto por al menos una instalación.
- *Continuous arc covering problem (CACP):* es una expansión de ACP en este método la cobertura de los arcos tiene que ser cubierta de manera completa y las instalaciones tienen que localizarse dentro de los mismos arcos.

En ReVelle, C.S.; Scholssberg, M.; Williamsa, J (2008) [15] se propone un método de resolución llamado HC 1-2-1, este método está pensado para aquellos en los que la programación lineal combinado con Branch & Bound sería muy difícil o consume demasiado tiempo para ser aplicado de manera regular. Este método es considerado metaheurístico porque parte de una solución heurística del problema y realiza un nuevo modelo heurístico con nuevas reglas para llegar a una solución mejor que la conseguida con la solución base.

Por último un documento interesante es el Brandeau, M.L. and Chiu, S.S (1989) [16] en el cual se hace un repaso general no exhaustivo de este campo, describiendo brevemente los diferentes tipos de modelos de localización que se habían estudiado hasta la fecha y se muestran las relaciones entre ellos.

3 ALGORITMO

El algoritmo explicado aquí es el método implícito del artículo [10]. En este modelo se realiza un proceso heurístico basado en el método GAS de Church and ReVelle [9], para intentar dar solución al problema de localización de estaciones base en un plano donde la demanda se representa mediante arcos.

Al contrario que otros modelos, permite flexibilidad para ajustar los pesos de la cobertura simple, doble y triple en el sistema. Además, en este modelo se considera que los puntos de demanda se encuentran repartidos uniformemente dentro de los arcos de demanda, siendo el objetivo de este algoritmo maximizar la demanda que es capaz de satisfacer. No se tiene en consideración que un usuario que esté usando los servicios pueda perder la conexión si se desplaza fuera del área de cobertura de la antena, al contrario que otros modelos cuyo objetivo es asegurar el mantemiento de una llamada de duración media de un punto de demanda que se desplaza por el arco.

Nuestro modelo es una aproximación puramente geométrica al problema. Su objetivo consiste en hallar, para cada arco, los segmentos de este con cobertura simple, doble o triple para calcular la demanda satisfecha. Para ello, se buscarán los puntos de cortes de las circunferencias de cobertura de las antenas con los arcos.

La función objetivo de nuestro sistema es:

$$\text{máx } f[(x_1, y_1), \dots, (x_n, y_n)] = \alpha_1 g_1[(x_1, y_1), \dots, (x_n, y_n)] + \alpha_2 g_2[(x_1, y_1), \dots, (x_n, y_n)] + \alpha_3 g_3[(x_1, y_1), \dots, (x_n, y_n)]$$

Donde:

- $(x_1, y_1), \dots, (x_n, y_n)$: son las coordenadas de las *n* estaciones a colocar.
- g_1, g_2, g_3: indican la fracción de la demanda cubierta para cada tipo de cobertura al colocarse las *n* estaciones base en las coordenadas indicadas por $(x_1, y_1), \dots, (x_n, y_n)$
- $\alpha_1, \alpha_2, \alpha_3$: son los pesos de las coberturas simple, doble y triple respectivamente.

Los distintos tipos de cobertura se muestran en la siguiente figura.

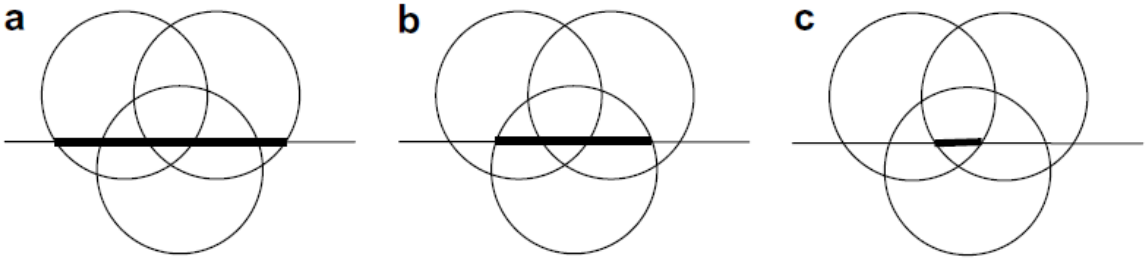

Figura 3-1

Los pesos de las coberturas serán siempre positivos para promover solapamiento entre las regiones, de tal forma que permita que las regiones con mayor demanda puedan tener respaldo suficiente. Sin embargo, es posible que cubrir una sola región con varias instalaciones no sea posible debido a restricciones del presupuesto, por lo tanto el proveedor debe mantener los pesos de las coberturas doble y triple dentro de un margen según sus intereses. Más adelante mostraremos las consecuencias en el resultado de asignar diferentes prioridades a las coberturas.

3.1 solución heurística

En este método, si se quieren colocar *n* antenas en el sistema, se tienen que fijar las posiciones de $n-1$ antenas y a continuación encontrar la posición óptima de la antena restante. Este proceso se tiene que realizar de forma secuencial hasta encontrar una solución sub-óptima para todas las antenas. Si asignamos un número a cada antena, desde 1 hasta *n*, el algoritmo comenzará fijando la posición de las antenas desde 2 hasta *n* y localizará la posición óptima de la antena 1 dentro de los posibles lugares. Después de colocar la antena 1 en su posición sub-óptima, se realiza el mismo procedimiento para la antena 2. El algoritmo continúa de esta forma hasta que todas las antenas sean colocadas.

Supongamos que dos instalaciones serán localizadas de forma que cubran un mismo arco de la red. La posición de una de las instalaciones es fija y cubre una longitud *x* de la longitud total del arco *l*. Si la segunda instalación se colocara justo encima de la primera, entonces la nueva función objetivo seria:

$$\alpha_1 \frac{x}{l} demanda + \alpha_2 \frac{x}{l} demanda = (\alpha_1 + \alpha_2)\frac{x}{l} * demanda$$

Si la segunda instalación fuera colocada en otro punto tal que cubre el mismo segmento *x* que la primera instalación y cubriera otro segmento adicional de longitud *y* en el arco, entonces la nueva función objetivo seria

$$\alpha_1 \left(\frac{x}{l} demanda + \frac{y}{l} demanda\right) + \alpha_2 \frac{x}{l} demanda = \alpha_1 \frac{y}{l} demanda + (\alpha_1 + \alpha_2)\frac{x}{l} demanda$$

Visualmente el grosor de cualquier camino es inversamente proporcional a la cantidad de cobertura dado a ese camino. Las figuras 3-2 y 3-3 son ejemplos de una red con 6 arcos en los que se localizarán 8 estaciones base. Asumiendo que el grosor inicial es el peso de la cobertura simple doble y triple $(\alpha_1 + \alpha_2 + \alpha_3)$, si colocamos una estación de tal forma que se aplique cobertura simple a un segmento del arco reduciremos el grosor del segmento en α_1.Reduciremos mas el grosor en α_2 si cubrimos el segmento con otra estación y el grosor será cero cuando el segmento este cubierto 3 o más veces.

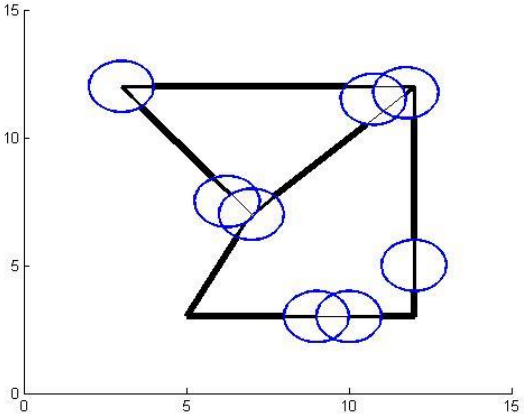

Figura 3-2

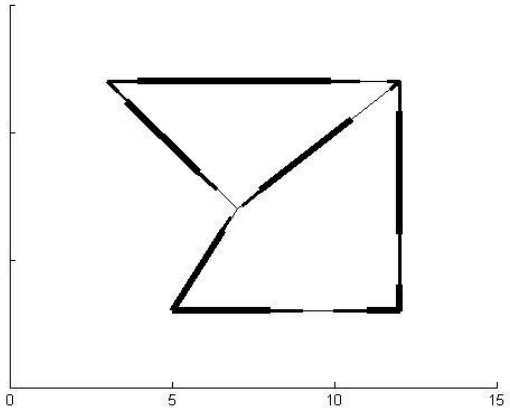

Figura 3-3

Asimismo consideramos que una estación base puede tener uno, dos o ningún punto de corte con cada uno de los arcos que conforman la red. Consideremos una red formada por 6 arcos tal como se muestra en la figura 3-4, la estación I no proporciona cobertura al sistema. La circunferencia de cobertura IV corta a los arcos en tres puntos, cada uno en un arco diferente. El círculo de cobertura III corta a un mismo arco en dos puntos. La circunferencia II corta a los arcos en cuatro puntos, dos puntos por cada arco. Asimismo en la codificación del protocolo también se ha añadido la posibilidad de que el área de una estación base no corte a un arco porque se encuentre totalmente incluido dentro de su área de cobertura.

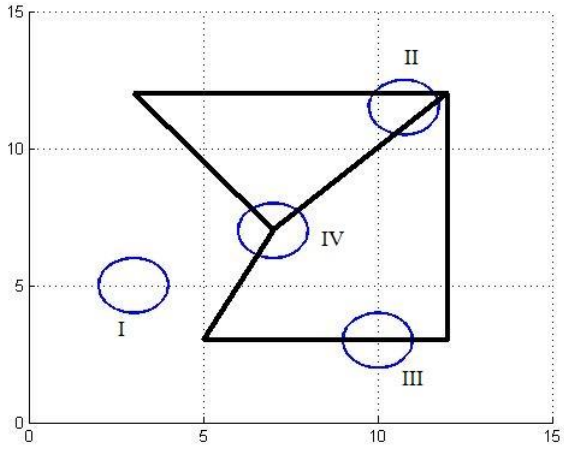

Figura 3-4

4 IMPLEMENTACIÓN

En este capítulo vamos a describir como hemos implementado el algoritmo en un programa para la realización de cálculos computacionales. Hemos realizado 4 códigos, el código base el cual hemos realizado basándonos en el algoritmo explicado en el artículo y 3 mejoras sobre este código que hemos considerado interesantes para ampliar sus funcionalidades.

4.1. Código base

La implementación del protocolo se ha hecho en MATLAB mediante el uso de dos funciones, *principal.m* que se encarga de realizar el algoritmo principal y *optimiza.m* que se encarga de calcular el resultado de la función objetivo.

La función principal toma como argumentos:

- X: matriz de $nx2$ elementos con las coordenadas origen y destino de los arcos de la red sobre el eje x. Donde n es el número de arcos del sistema.
- Y: matriz de $nx2$ elementos con las coordenadas origen y destino de los arcos de la red sobre el eje y. Donde n es el número de arcos del sistema.
- Demand: vector de n elementos con las demandas de los arcos
- Radio: longitud del radio de cobertura de las estaciones base
- Single: peso de la cobertura simple
- Double: peso de la cobertura doble
- Triple: peso de la cobertura triple
- nBS: número de estaciones base a colocar
- xBS (opcional): vector inicial con las coordenadas de las N-1 estaciones base.
- yBS (opcional): vector inicial con las coordenadas de las N-1 estaciones base.

Un ejemplo de los parámetros para generar el escenario de la figura 4-1 sería: principal([1 14; 1 8], [1 1; 1 14], [60 90], 2, 100, 0, 0, 2, [5 10], [9 2])

A partir de las matrices X e Y se genera un plano limitado por los valores máximos y mínimos de ambas matrices. A continuación generamos sobre el plano puntos equiespaciados que serán las posibles localizaciones de las estaciones base. En la figura 4-1 se muestra un escenario de ejemplo en el que tenemos 2 arcos y hemos definido la distancia entre los puntos potenciales como 0.5 unidades del plano.

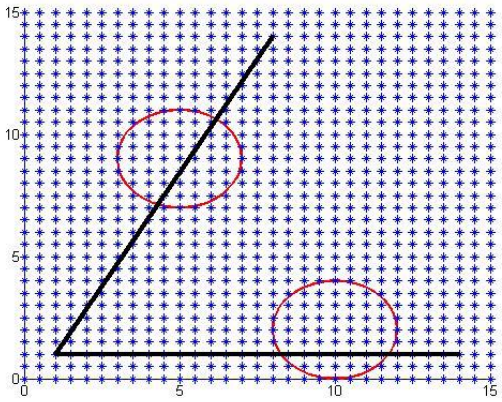

Figura 4-1

Para la realización de este método necesitamos partir de una solución inicial con $N-1$ estaciones base, para ello utilizamos los argumentos xBS e yBS. Sin embargo, si se llama a la función sin estos parámetros o la longitud del vector no es $N-1$, se advertirá al usuario y se le preguntará como desea introducir la solución inicial. Los métodos implementados son:

- Introducción de un vector xBS y un vector yBS: Se le pedirá al usuario que introduzca un vector de Matlab de $N-1$ elementos primero para el vector xBS y luego para el vector yBS, en el caso de introducir un vector de diferentes dimensiones saltará un error y terminará el programa. Una vez introducidos los vectores de forma correcta el programa colocará las estaciones en los puntos potenciales más cercanos a las coordenadas introducidas y luego continuará automáticamente.
- Introducción gráfica de las estaciones base: Se generará una figura con el plano de la red y se le pedirá al usuario que marque $N-1$ puntos sobre este mapa. Con este método no cabe la posibilidad de que el usuario marque un número menor o mayor de puntos. Una vez introducida la cantidad necesaria el programa colocará las estaciones en los puntos potenciales más cercanos a las coordenadas introducidas y luego continuará automáticamente.
- Solución aleatoria: si el usuario elige esta opción se generaran dos vectores aleatorios de manera independiente donde los resultados estarán comprendidos entre el mínimo y el máximo valor de las coordenadas de los arcos del plano.

Cabe resaltar que aunque en el algoritmo principal no se contempla la posibilidad de que dos estaciones base sean colocadas en un mismo punto del plano, en la solución inicial si se permite esta posibilidad ya que mediante la ejecución del programa se corregirá este aspecto.

Como hemos dicho este método tiene un enfoque puramente geométrico, por lo que necesitaremos las ecuaciones de las rectas que representan los arcos y las circunferencias centradas en las coordenadas de las estaciones base que representan los radios de las antenas.

Para conseguir esto calculamos primero las pendientes de la rectas como la diferencia entre las coordenadas origen y destino del eje y de las rectas por cada unidad en el eje x. Una vez conseguida la pendiente calculamos el punto de corte con el origen usando la ecuación de la recta.

$$y_1 = mx + n$$

Estos valores los guardamos en los vectores pendiente y origen respectivamente. Si quisiéramos acceder a toda la información del arco *i* solo tendríamos que hacer: x(i,1), x(i,2), y(i,1), y(i,2), demand(i), pendiente(i), origen(i).

A continuación tenemos que calcular los puntos de cortes de las circunferencias de cobertura de las antenas con cada uno de los arcos del plano, para ello hacemos uso de un bucle anidado de tal forma que, para cada estación *k*, se calcule los puntos de cortes con todos los arcos. Para llevar a cabo esto calculamos la semicircunferencia superior e inferior de la estación base en cada iteración de *k*, haciendo uso de la siguiente ecuación.

$$y_2 = \pm\sqrt{r^2 - (x_{aux} - a)^2} + b$$

Donde x_{aux} es un vector auxiliar de 1000 unidades que va desde $xBS(k) - radio$ hasta $xBS(k) + radio$, a es el valor $xBS(k)$ y b es el valor $yBS(k)$. Calculamos así la semicircunferencia superior e inferior de la antena.

Una vez conseguidas ambas semicircunferencias, vamos generando iteración a iteración el vector y_{aux} del arco *i* haciendo uso de la ecuación de la recta usando el mismo vector auxiliar x_{aux}. Para averiguar las coordenadas de los puntos de corte restamos tanto a la semicircunferencia superior como a la inferior los valores de y_{aux} , hallando la posicion *p* del primer elemento del vector solución que es cero o cambia de signo y guardamos en las matrices cortesX y cortesY los valores de x_{aux}(p) e y_2(p) respectivamente. Se tiene en cuenta que ambos puntos de corte pueden pertenecer a una misma semicircunferencia, por lo cual se comprobará que no haya más cambios de signos. Recordemos que una estación base puede tener cero, uno o dos cortes con cada arco.

Al generar los vectores y_{aux} del arco mediante la ecuación de la recta y el vector x_{aux}, los puntos de corte encontrados podrían no estar incluidos en el arco. Dicho esto una vez tenemos los puntos de corte, en el caso que tengamos uno o dos puntos, comprobamos que éstos están dentro del rango origen y destino del arco, podemos realizar la comprobación o bien con las coordenadas Y o con las X. No hacen falta comprobar ambas puesto que se han calculado los puntos usando la ecuación de la recta. Se pueden dar los siguientes casos:

- Ambos puntos de cortes están dentro del rango origen y destino: significa que ambos puntos de corte forman parte de un sub-segmento del arco.
- Sólo un punto de corte está dentro del rango origen y destino: significa que la circunferencia de la estación sólo corta al arco en un punto, por lo tanto procederemos a cambiar el punto de corte que no está incluido en el arco por la coordenada origen o destino que se encuentren dentro del área de la circunferencia.
- Ninguno de los puntos están dentro del rango. De aquí se derivan dos opciones:
    - El arco se encuentra fuera del área de cobertura de la antena. En este caso se borrarán los valores introducidos en las matrices cortesX y cortesY.
    - El arco es lo suficientemente pequeño como para estar incluido en su totalidad en el área de cobertura de la antena. En este caso los puntos de cortes serán los puntos origen y destino del arco.

Una vez rellenadas las matrices cortesX y cortesY hallamos la solución de la función objetivo, para ello llamamos a la función *optimiza.m.*

La función optimiza.m tiene los siguientes argumentos:

- cortesX: matriz con las coordenadas X de los puntos de corte de las circunferencias de cobertura de las estaciones con los arcos

- cortesY: matriz con las coordenadas Y de los puntos de corte de las circunferencias de cobertura de las estaciones con los arcos
- single - peso de la cobertura simple
- double - peso de la cobertura doble
- triple - peso de la cobertura triple
- demand - vector con las demandas de los arcos

Esta función coge los puntos de corte de un arco y calcula la proporción del arco con cobertura simple, doble y triple y halla la demanda cubierta usando la siguiente ecuación

$$resultado = \sum_{k=1}^{K} \sum_{i=1}^{3} \frac{X_{ik}}{l_k} * \alpha_i * demand_k$$

Siendo X_{ik} la longitud del arco *k* cubierta con cobertura *i*, l_k la longitud total del arco *k*, α_i el peso de las cobertura simple, doble y triple y $demand_k$ la demanda del arco *k*.

El doble sumatorio se corresponde a dos bucles anidados como los utilizados para el cálculo de los puntos de cortes. La proporción X_{ik}/l_k se ha calculado usando la proyección sobre los ejes del segmento formado por los puntos de cortes con respecto a la proyección del arco total y utilizando un vector auxiliar para distinguir las zonas con distinto tipo de cobertura.

Una vez obtenido el valor de la función optimiza.m se tiene la solución inicial completa y estamos preparados para realizar el cuerpo del algoritmo. Se ha implementado usando un triple bucle anidado de tal forma que las nBS estaciones recorran todos los puntos de la matriz formada por las coordenadas x e y de las localizaciones potenciales.

En cada paso del algoritmo la estación k comprueba que en el siguiente punto sobre el que se va a colocar no haya ninguna estación, en cuyo caso cambia sus coordenadas y calcula sus puntos de cortes y llama a optimiza.m siendo dos posibles situaciones: o bien el resultado obtenido es mejor que el guardado, en cuyo caso guarda la posición actual como la optima temporal, las matrices de los puntos de corte y el nuevo resultado como el optimo temporal, o en caso contrario avanza al siguiente punto.

Una vez que una estación base ha recorrido todos los puntos potenciales guarda los óptimos temporales como definitivos y es el turno de la siguiente estación.

Una vez que todas las estaciones han sido colocadas tenemos la solución al problema de localización y el programa dibuja una figura con la red, los puntos sobre los cuales se situarán las antenas y el radio de alcance de todas, además en los argumentos de salida se devuelven las coordenadas de las estaciones base (xBS,yBS), el resultado de la optimización y el porcentaje conseguido.

La siguiente figura muestra un diagrama de flujo del sistema.

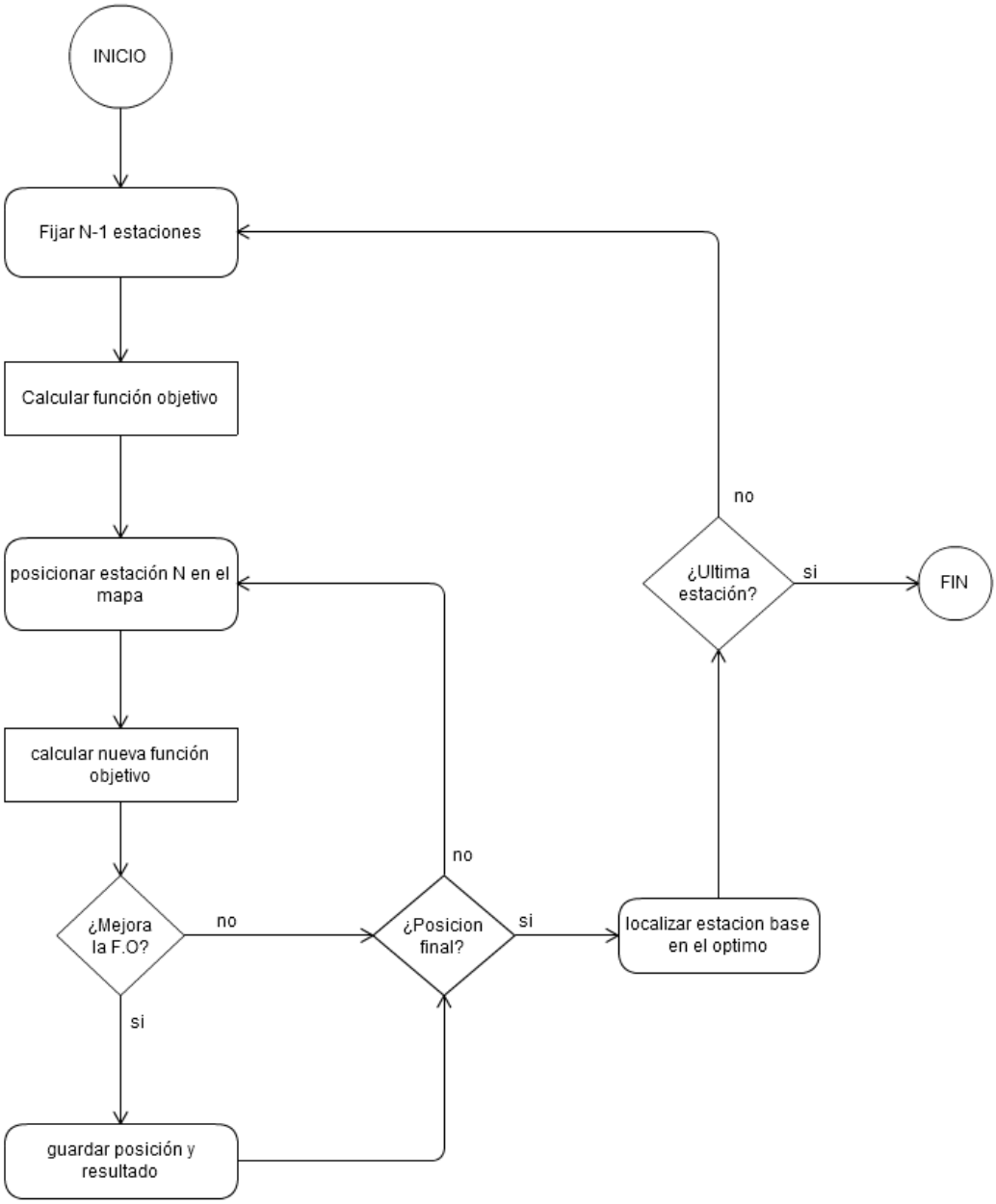

Figura 4-2

4.2. Mejoras

Hemos añadido diferentes versiones con mejoras para ampliar la funcionalidad y usos al código base, como la posibilidad de usar diferentes estaciones base que se diferenciarán en el radio de alcance de la antena. Sobre esta mejora añadimos la posibilidad de que el operador tenga ya construidas estaciones base sobre el mapa, en este caso esas estaciones aportan cobertura que hay que tener en cuenta y suponemos el coste de traslado o desmontaje es alto.

También hemos añadido la posibilidad de que el operador quiera asegurar una mínima demanda al finalizar la simulación y ver el coste y número final de estaciones base necesarias. En este caso una vez alcanzada la solución con el número de estaciones base correspondiente se analizará si se cumple el porcentaje de cobertura especificado y en caso contrario añadirá nuevas estaciones hasta que se alcance el objetivo.

4.2.1. Diferentes tipos de estaciones

La mejora que hemos incluido en esta versión del programa es la posibilidad de usar estaciones base que se diferencien entre sí en el radio de cobertura que son capaces de suministrar. En cuanto a telefonía móvil, ésta tiene una estructura celular que se dividen entre macro células (15-20Km) para entornos rurales, micro células (1Km) y pico células (<1Km) para entornos urbanos y suburbanos, este radio también varía según el espectro de frecuencias que utiliza, así para una antena 4g es diferente a la red normal instalada, y a su vez la altura de la antena y el entorno también afectan mediante reflexiones, interferencias u obstáculos.

En cuanto a antenas wifi este radio varía según la marca de la antena que usemos, la potencia que se le dé, la frecuencia que utilice y también a su vez el entorno, las interferencias, los cuerpos que estén por medio, las características de estos cuerpos... etc.

La implementación en el código se ha hecho modificando el parámetro *radio* de la función *principal.m* para que sea un vector de *n* elementos siendo *n* el número de antenas en el sistema, de forma que para acceder al radio de la antena i, basta con acceder al elemento i de ese vector.

4.2.2. Estaciones fijas

En las anteriores versiones del código se consideraba un despliegue total de la red para un operador nuevo, sin embargo la situación normal es que un operador tenga que aumentar el número de estaciones base dentro de un territorio debido al incremento de la demanda de servicios o a la aparición de nuevas tecnologías. O, en el caso de un operador nuevo, que le compre parte de los servicios a otra operadora.

En esta versión añadimos la posibilidad de colocar antenas fijas en el mapa. Estas antenas proporcionan cobertura a la red y se consideran fijas, por lo tanto no estarán condicionadas a los lugares potenciales de las nuevas antenas ni entrarán en el algoritmo de colocación en los puntos candidatos, ya que se considera que el coste de mover una torre de telecomunicaciones es demasiado elevado.

Para implementar esta mejora se han tenido que añadir 3 parámetros a la función principal.m, esos parámetros son:

- xBSfijos: coordenadas del eje *x* sobre el plano de nuestro sistema de las *n* estaciones fijas. Vector de 1x*n* elementos.
- yBSfijos: coordenadas del eje *y* sobre el plano de nuestro sistema de las *n* estaciones fijas. Vector de 1x*n* elementos.
- radiofijos: vector de 1x*n* elementos que indican el radio de cobertura de las *n* estaciones base fijas

La implementación de esta mejora requiere que en la solución inicial del problema cuente con la cobertura dada por estas estaciones fijas, para ellos hemos utilizado el código ya implementado en el código base para calcular los puntos de corte de los radios de cobertura de las estaciones base con los arcos y añadirlos a la matriz cortesx y cortesy para que, cuando se llame a la función *optimiza.m* para calcular el resultado inicial, cuente con las nuevas antenas.

Esta versión incluye la anterior versión, por lo que tanto las nuevas estaciones como las fijas podrán tener radios distintos.

4.2.3. Porcentaje de demanda

Uno de los intereses que tiene nuestra aplicación desde el punto de vista de un operador es asegurar que se satisface un cierto porcentaje de la demanda. Entre otras cosas para comprobar que la solución suministrada por el programa cumpla con las metas deseadas. Por ello nuestra última mejora a nuestro código considera que, si a partir de una solución inicial, se encuentra una solución factible que no cumple los requisitos del operador, se irán añadiendo tantas estaciones base como se necesite para cumplir con el objetivo o bien se alcance un valor crítico de antenas.

Para la implementación de esta mejora se han añadido dos parámetros a la llamada de la función principal.m, estos son:

- porcentaje: porcentaje sobre la demanda total del sistema que se desea asegurar.
- exceso: número máximo de estaciones base adicionales que el operador consideraría instalar

Una vez terminado el código base, en este caso el código explicado en el apartado anterior, se comprueba la demanda satisfecha, si es menor que el pasado por parámetro se añade una nueva estación base, la cual se colocará en la posición disponible donde maximice la función objetivo, siendo el proceso para calcular el resultado el mismo que el empleado en las anteriores versiones del código. Este proceso se repetirá hasta que el porcentaje requerido sea cumplido o hasta que se alcance el número máximo de estaciones base adicional.

Cabe destacar que una vez llegado a esta parte del programa el algoritmo empleado sería el equivalente a usar el algoritmo Greedy Addition durante la iteración *n*. Este algoritmo, explicado anteriormente, fue expuesto por Church and ReVelle [9] y consiste en la adición progresiva de instalaciones sobre un plano en la posición óptima donde se cubra más cobertura.

Como hemos dicho esta versión incluye las anteriores, sin embargo aunque tanto para las estaciones fijas como las nuevas se permite que tengan radios de alcance variables, se considera que las antenas adicionales a instalar en caso de no cumplirse el umbral de la demanda tienen todas el mismo radio, siendo este el máximo de entre las estaciones nuevas a colocar, esto se ha tomado así ya que el objetivo llegados a este punto cubrir la máxima cobertura posible con el menor número de estaciones adicionales.

5 CÁLCULO COMPUTACIONAL

En este capítulo realizaremos una serie de simulaciones sobre un escenario para demostrar el comportamiento del programa y analizar los distintos resultados obtenidos en consecuencia de los parámetros suministrados. Todas las simulaciones se han realizado con la versión R2013b de MATLAB.

El escenario será el mostrado en la figura 5-1, que se codificará en los parámetros *x*, *y* y *demanda* de la función *principal.m*. El resto de parámetros dependerán de la simulación, se realizarán diferentes casos variando los pesos de la cobertura simple, doble y triple, y por cada caso se realizarán dos experimentos, uno en el que las posiciones de las estaciones base se proporcionarán de forma aleatoria y otro en el que las posiciones sean proporcionadas al comienzo y sean las mismas en todas las simulaciones.

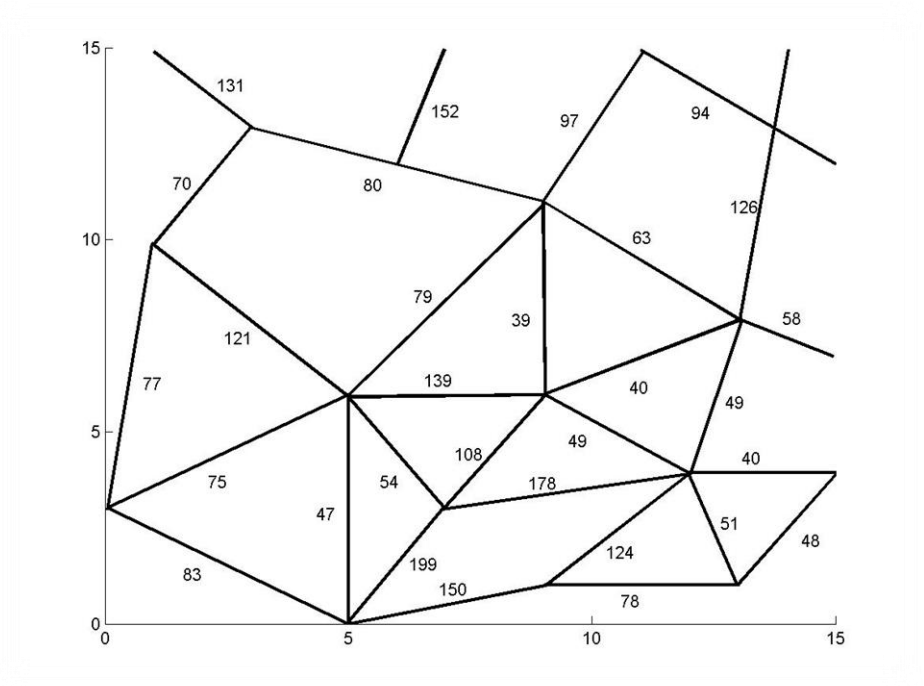

Figura 5-1

La figura tiene un tamaño de 15 unidades por 15 unidades, el mallado de los puntos potenciales de colocación de estaciones base se ha considerado cada 0.5 unidades tanto en horizontal como en vertical. El radio de las antenas se ha considerado por defecto de 2 unidades.

Se realizarán 4 casos para cada uno de los códigos desarrollados en los que se variarán los pesos de las coberturas. Siendo para los cuatro casos los siguientes pesos de las coberturas simple, doble y triple respectivamente: (100, 0,0), (70, 20,10), (20, 60,20), (0, 0,100). A su vez, para cada caso se partirá de una solución inicial aleatoria y una solución inicial suministrada de manera igual para todos los casos.

5.1 Código base

En este apartado usaremos la solución inicial mostrada en la figura 5-2 para todos los casos.

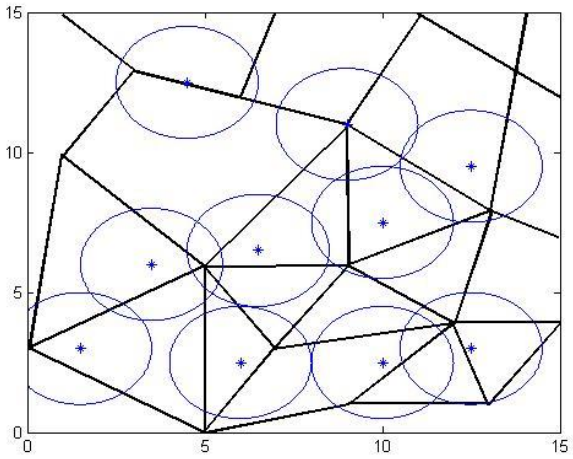

Figura 5-2

5.1.1 Caso 1

En este caso estudiaremos los resultados computacionales del código base en nuestro escenario si los pesos de las coberturas simple, doble y triple son 100, 0 y 0 respectivamente.

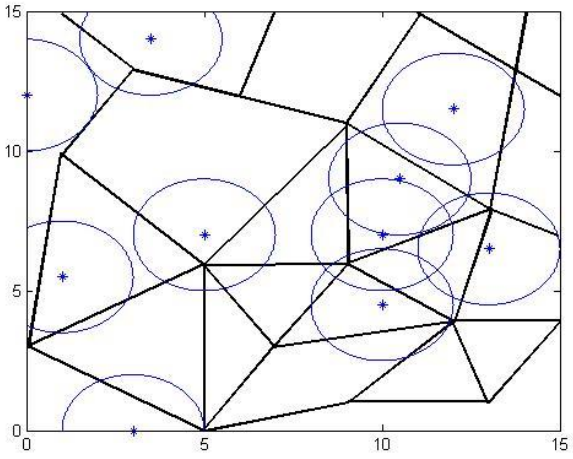

Figura 5-3

La figura 5-3 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-4 y 5-5 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente

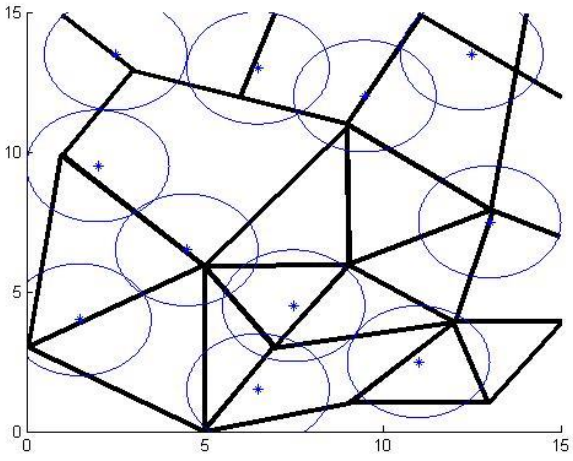

Figura 5-4

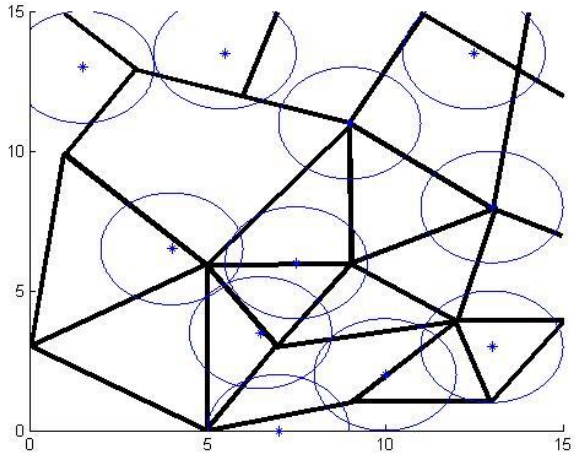

Figura 5-5

5.1.2 Caso 2

En este caso estudiaremos los resultados computacionales del código base en nuestro escenario si los pesos de las coberturas simple, doble y triple son 70, 20 y 10 respectivamente.

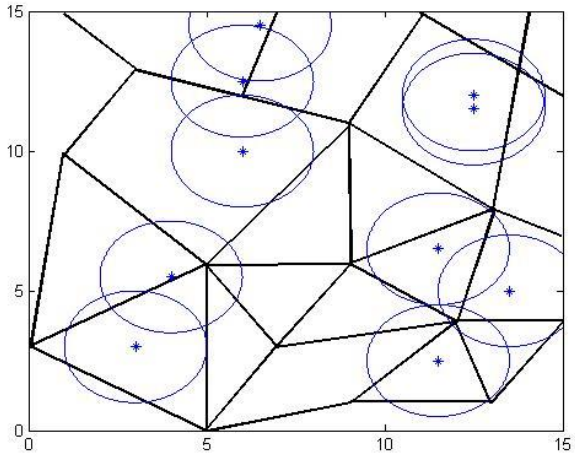

Figura 5-6

La figura 5-6 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-7 y 5-8 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

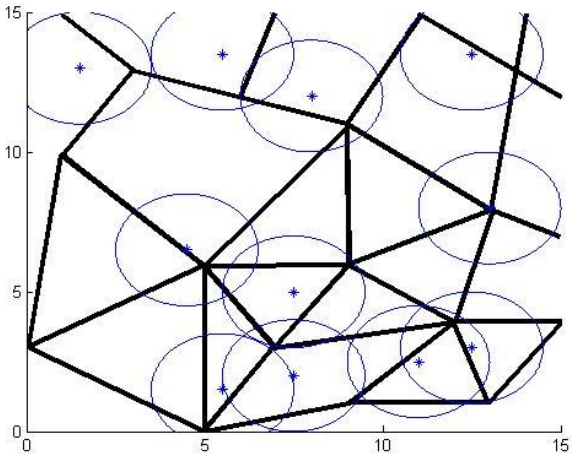

Figura 5-7

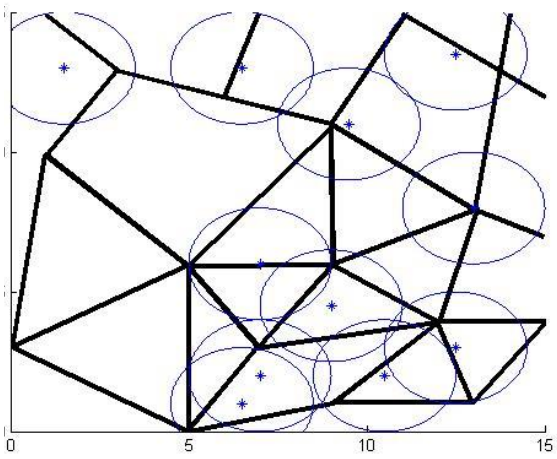

Figura 5-8

5.1.3 Caso 3

En este caso estudiaremos los resultados computacionales del código base en nuestro escenario si los pesos de las coberturas simple, doble y triple son 20, 60 y 20 respectivamente.

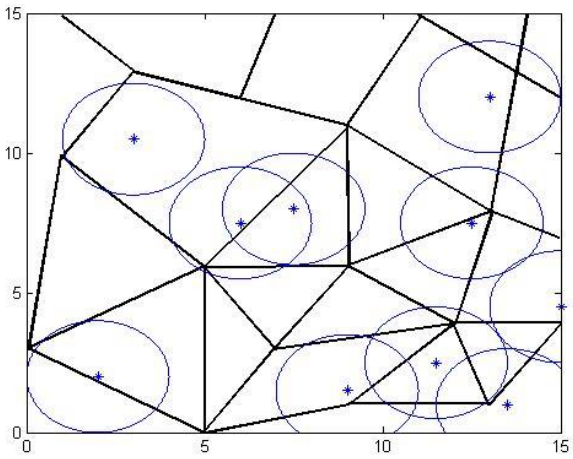

Figura 5-9

La figura 5-9 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-10 y 5-11 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

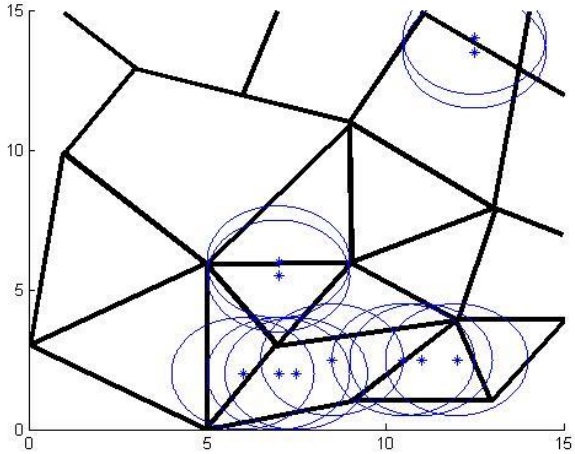

Figura 5-10

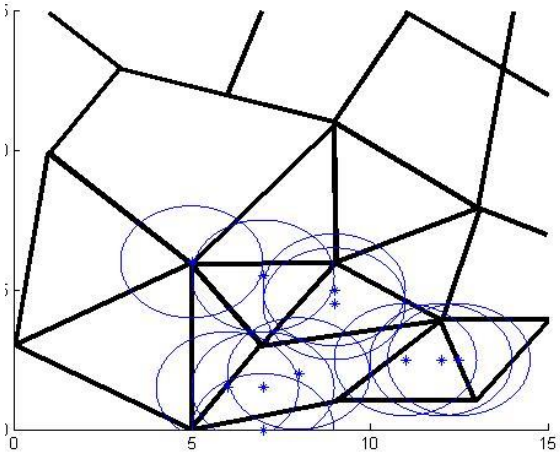

Figura 5-11

5.1.4 Caso 4

En este caso estudiaremos los resultados computacionales del código base en nuestro escenario si los pesos de las coberturas simple, doble y triple son 0, 0 y 100 respectivamente.

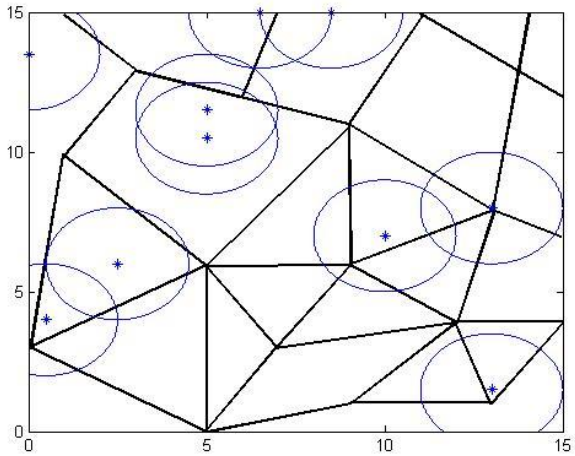

Figura 5-12

La figura 5-12 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-13 y 5-14 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

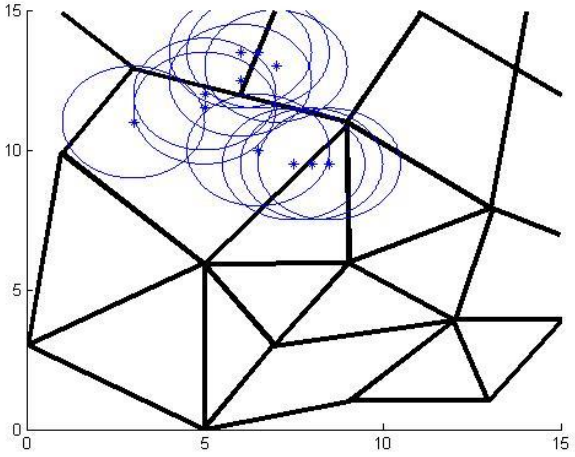

Figura 5-13

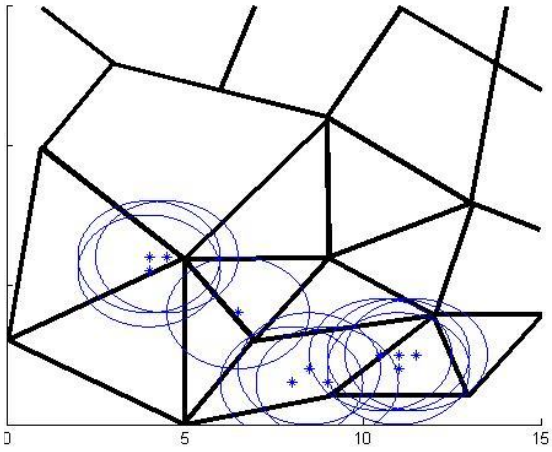

Figura 5-14

5.2 Mejora 1: diferentes radios

En este apartado habrá 3 estaciones base con radio 1 unidad, 4 estaciones base con radio 2 unidades y 4 estaciones base con radio 2,5 unidades. La solución inicial usada como constante en todos los casos será la mostrada en la figura 5-15.

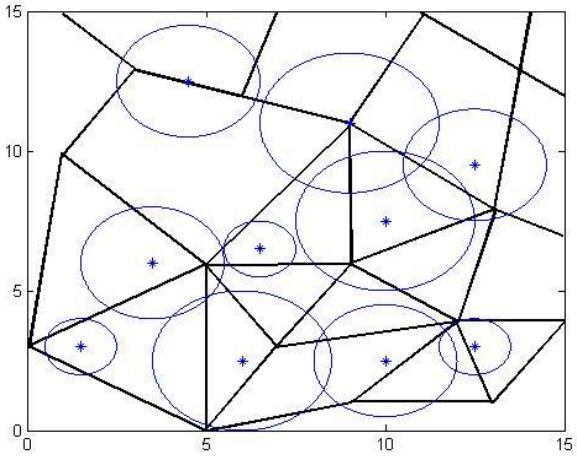

Figura 5-15

5.2.1 Caso 1

En este caso estudiaremos los resultados computacionales del código con la mejora 1 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 100, 0 y 0 respectivamente.

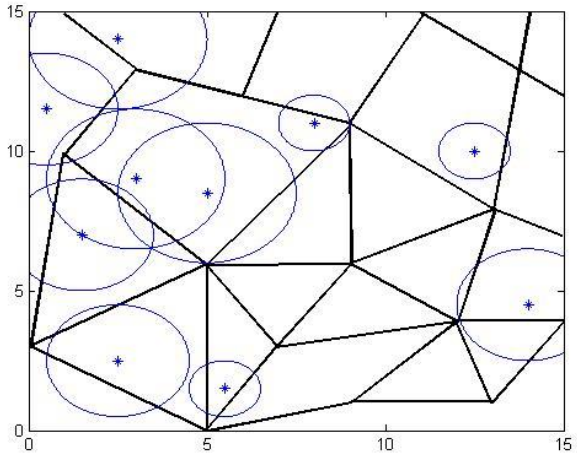

Figura 5-16

La figura 5-16 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-17 y 5-18 muestran las soluciones finales de las soluciones iniciales aleatoria y fija respectivamente.

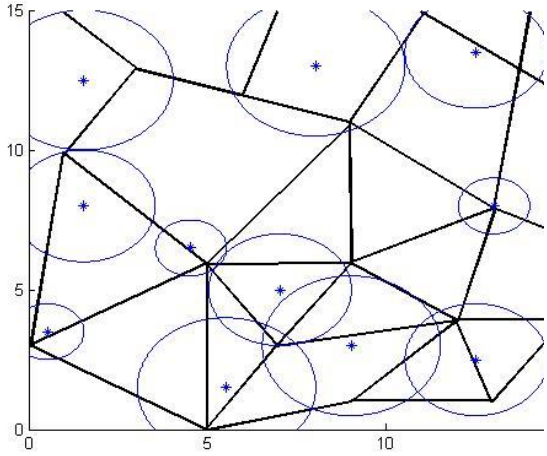

Figura 5-17

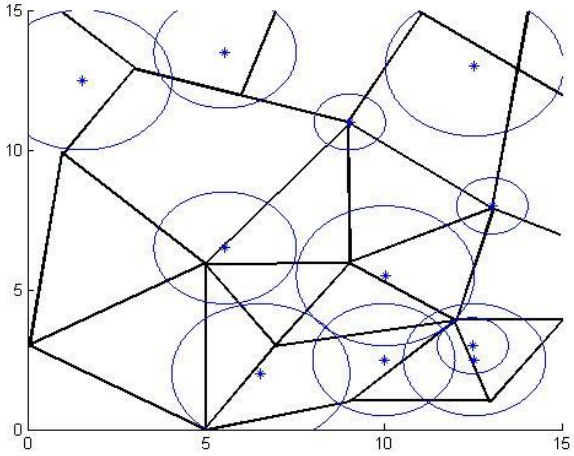

Figura 5-18

En la figura 5-18 podemos observar que en la esquina inferior derecha hay una estación base de radio 1 que no aporta nada al sistema ya que en este escenario solo se busca cobertura simple en todo el mapa. Esto se debe al problema ya comentado por Curch and ReVelle [9] en su modelo GA al decir que a medida que el algoritmo avanza puede que la posición de una estación ya no esté justificada.

5.2.2 Caso 2

En este caso estudiaremos los resultados computacionales del código con la mejora 1 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 70, 20 y 10 respectivamente.

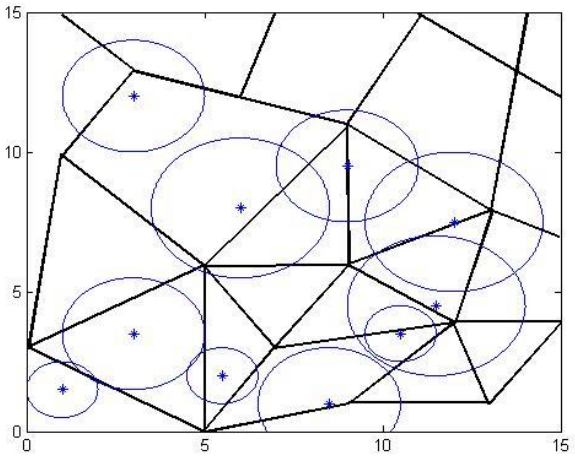

Figura 5-19

La figura 5-19 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-20 y 5-21 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

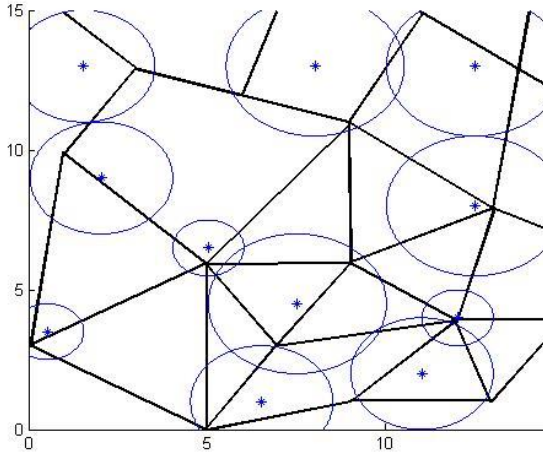

Figura 5-20

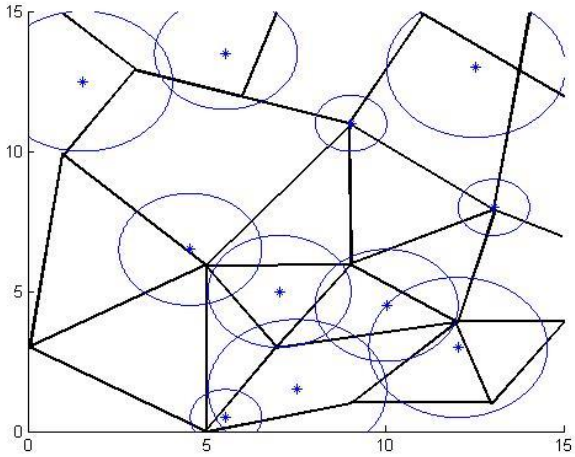

Figura 5-21

5.2.3 Caso 3

En este caso estudiaremos los resultados computacionales del código con la mejora 1 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 20, 60 y 20 respectivamente.

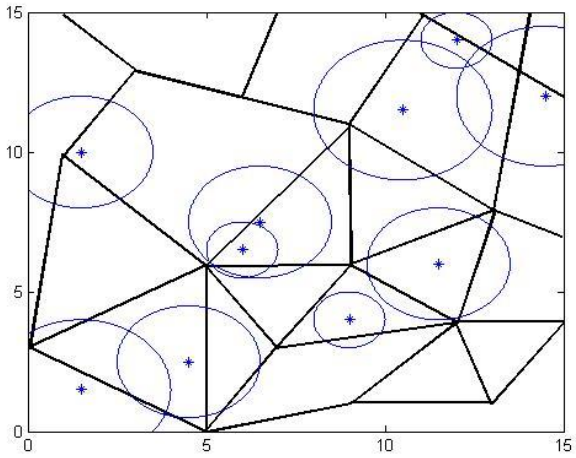

Figura 5-22

La figura 5-22 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-23 y 5-24 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

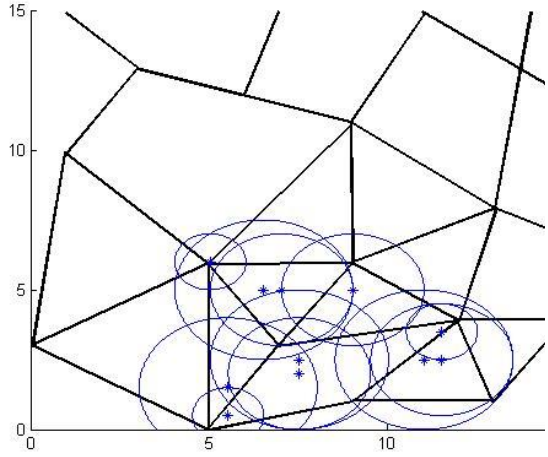

Figura 5-23

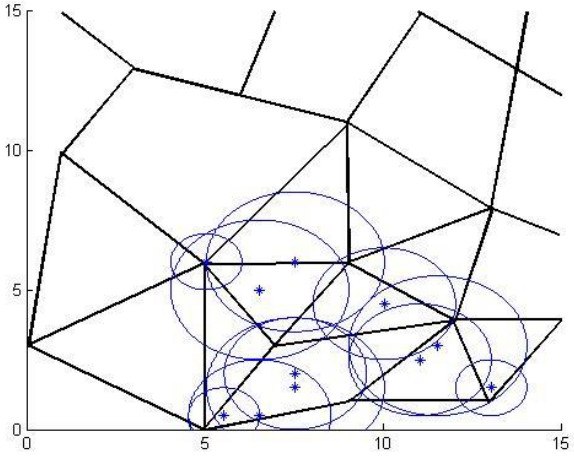

Figura 5-24

5.2.4 Caso 4

En este caso estudiaremos los resultados computacionales del código con la mejora 1 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 0, 0 y 100 respectivamente.

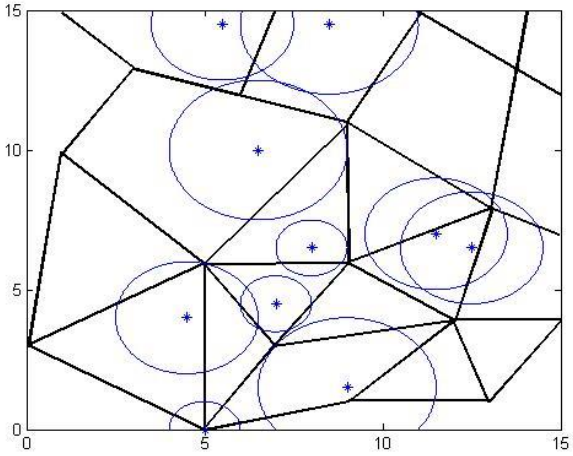

Figura 5-25

La figura 5-25 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-26 y 5-27 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

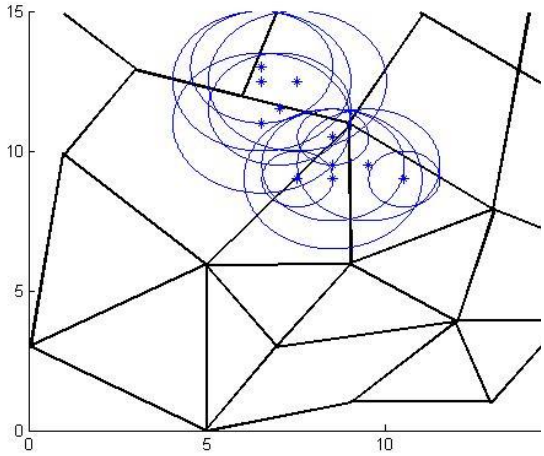

Figura 5-26

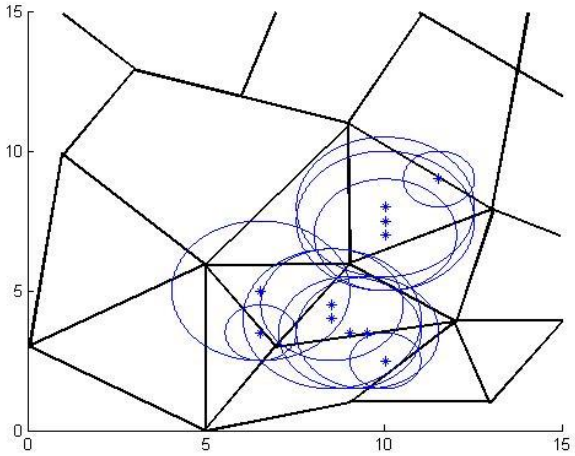

Figura 5-27

5.3 Mejora 3: Estaciones fijas

En este apartado habrá 4 estaciones base fijas que se cogerán de las 11 estaciones del caso anterior, se usarán dos con radio 1 unidad, una con radio 2 unidades y una con radio 2.5 unidades, la solución inicial usada como constante en todos los casos es la mostrada en la figura 5-28, siendo las estaciones fijas las de color rojo.

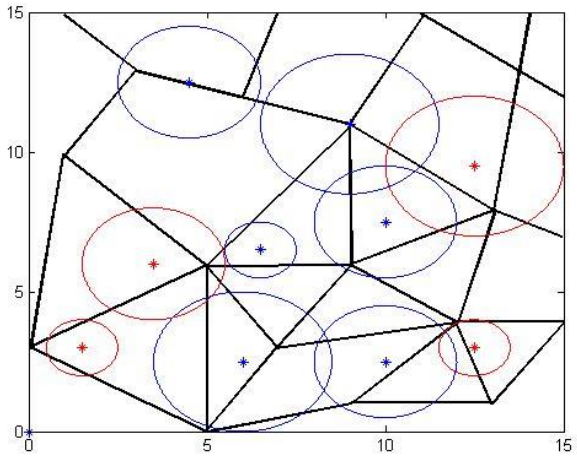

Figura 5-28

5.3.1 Caso 1

En este caso estudiaremos los resultados computacionales del código con la mejora 2 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 100, 0 y 0 respectivamente.

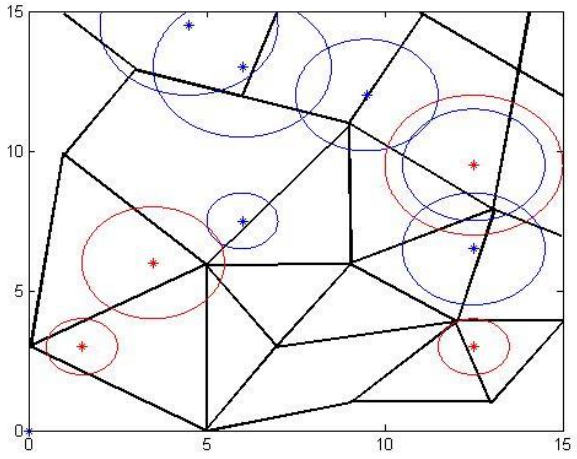

Figura 5-29

La figura 5-29 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-30 y 5-31 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

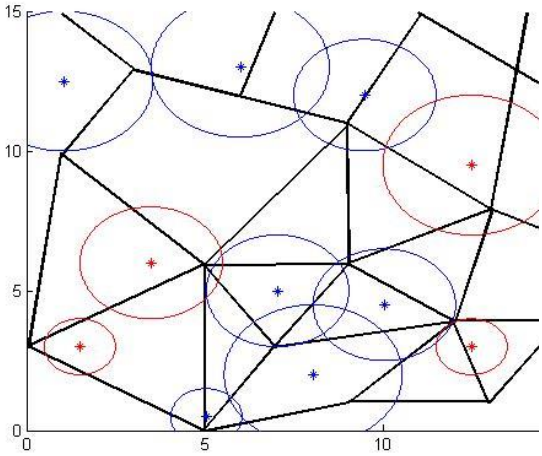

Figura 5-30

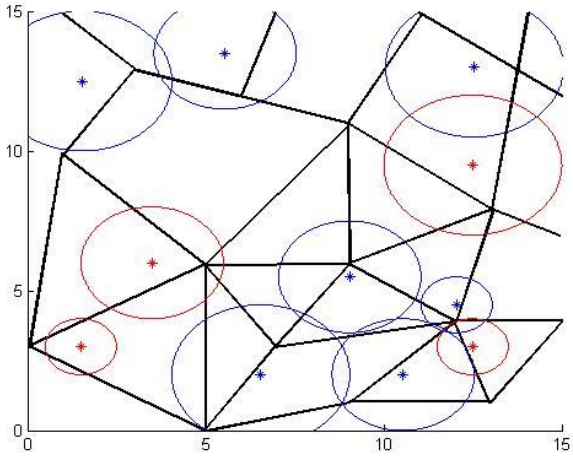

Figura 5-31

5.3.2 Caso 2

En este caso estudiaremos los resultados computacionales del código con la mejora 3 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 70, 20 y 10 respectivamente.

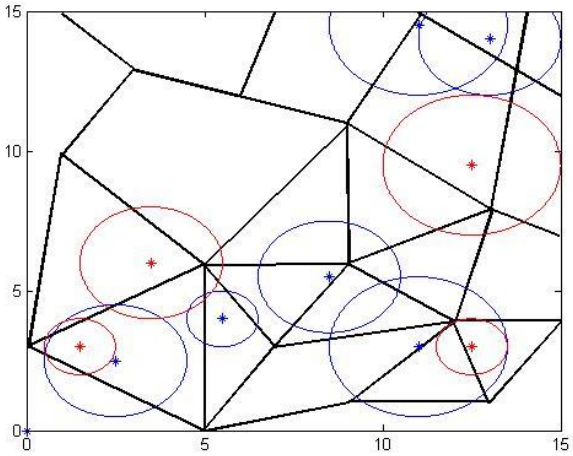

Figura 5-32

La figura 5-32 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-33 y 5-34 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

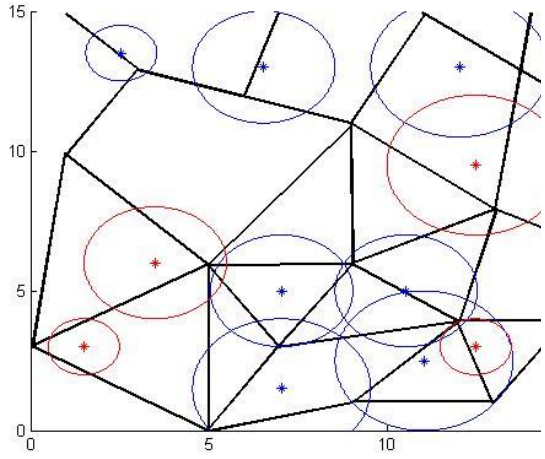

Figura 5-33

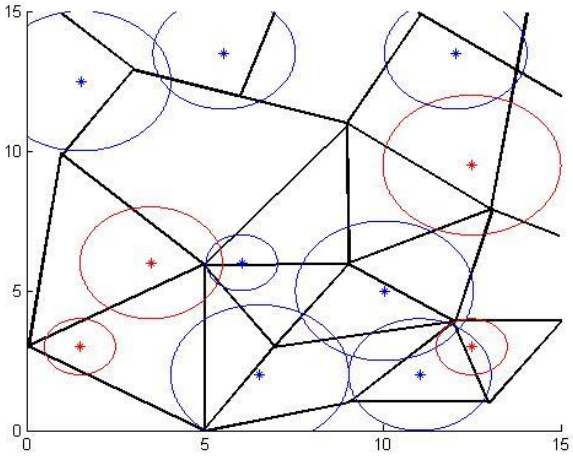

Figura 5-34

5.3.3 Caso 3

En este caso estudiaremos los resultados computacionales del código con la mejora 3 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 20, 60 y 20 respectivamente.

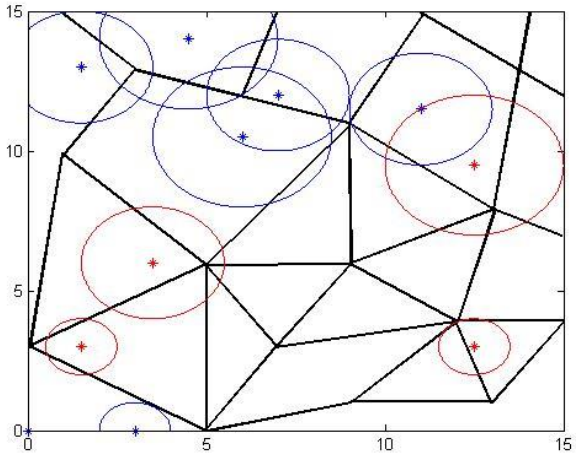

Figura 5-35

La figura 5-35 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-36 y 5-37 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

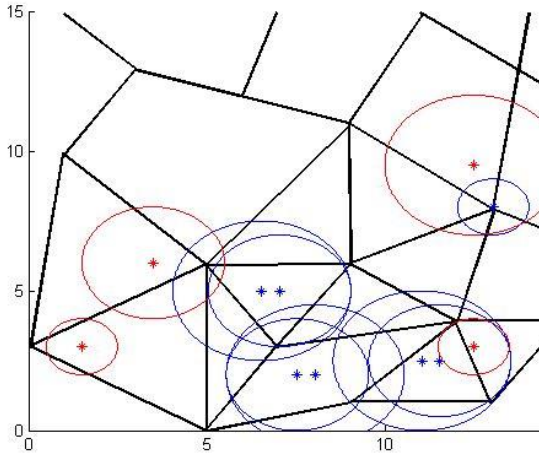

Figura 5-36

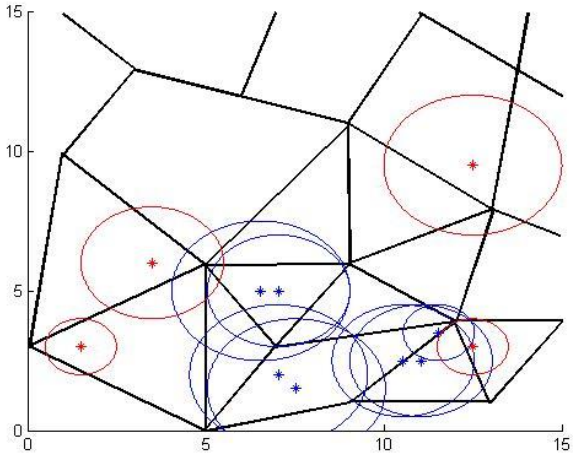

Figura 5-37

5.3.4 Caso 4

En este caso estudiaremos los resultados computacionales del código con la mejora 3 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 0, 0 y 100 respectivamente.

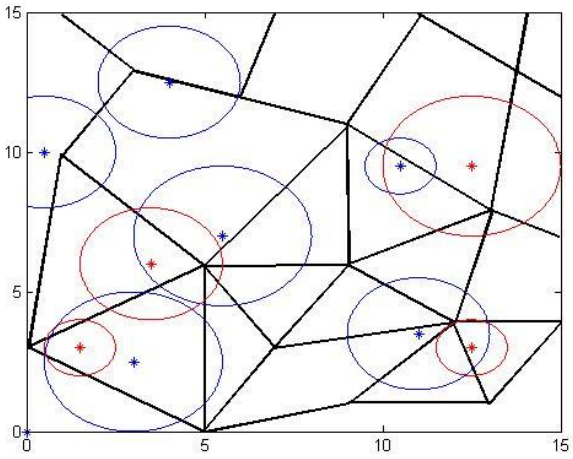

Figura 5-38

La figura 5-38 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-39 y 5-40 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

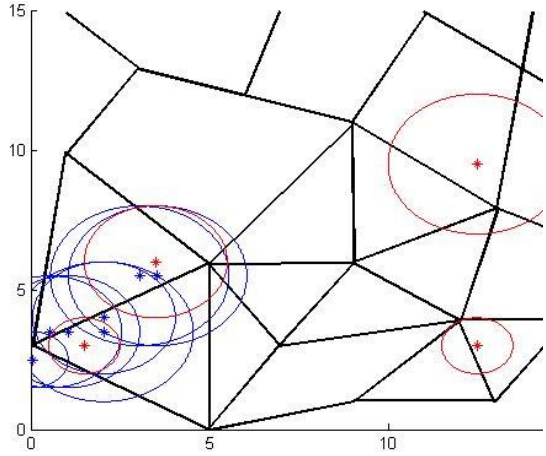

Figura 5-39

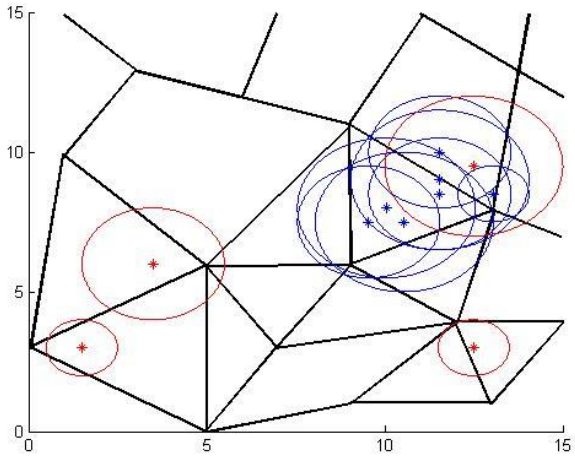

Figura 5-40

5.4 Mejora 4: Porcentaje de demanda

En este apartado se irá cambiando el porcentaje de demanda requerido en cada caso para obtener un resultado viable sin tener que añadir un número elevado de estaciones base nuevas. El número máximo de estaciones base nuevas a colocar se ha establecido en 5. La figura 5-41 muestra la solución inicial usada como constante en todos los casos. Las nuevas estaciones base añadidas en las soluciones serán representadas de color verde.

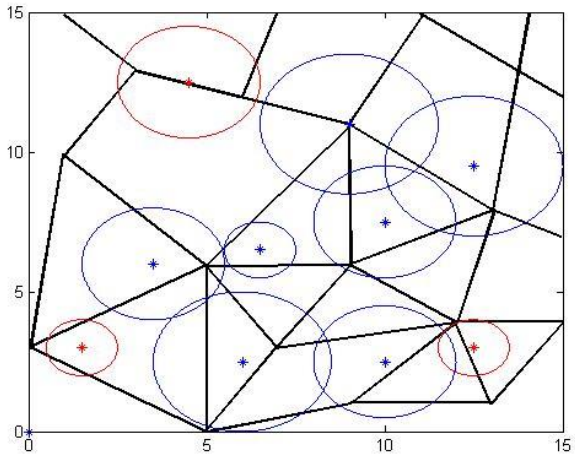

Figura 5-41

5.4.1 Caso 1

En este caso estudiaremos los resultados computacionales del código con la mejora 4 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 100, 0 y 0 respectivamente. El porcentaje de demanda requerido se ha puesto al 80%.

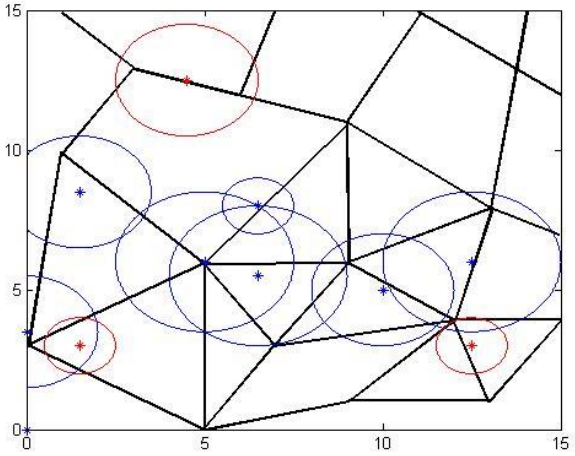

Figura 5-42

La figura 5-42 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-43 y 5-44 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

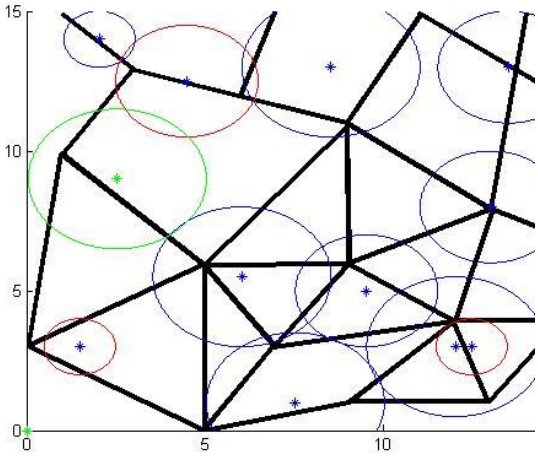

Figura 5-43

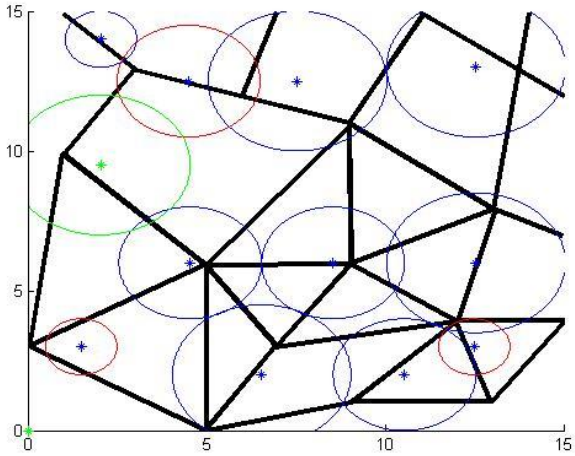

Figura 5-44

5.4.2 Caso 2

En este caso estudiaremos los resultados computacionales del código con la mejora 4 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 70, 20 y 10 respectivamente. El porcentaje de demanda requerido se ha puesto al 65%.

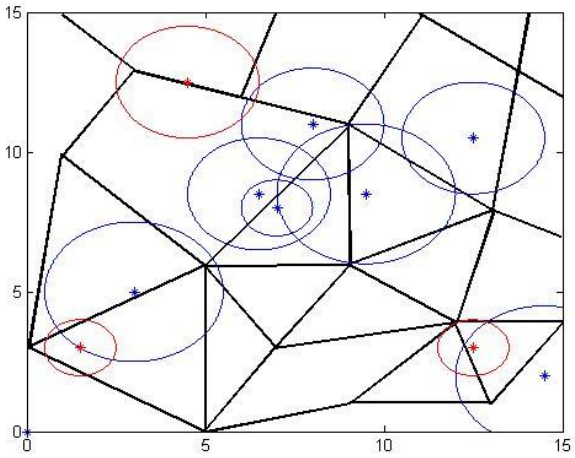

Figura 5-45

La figura 5-45 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-46 y 5-47 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

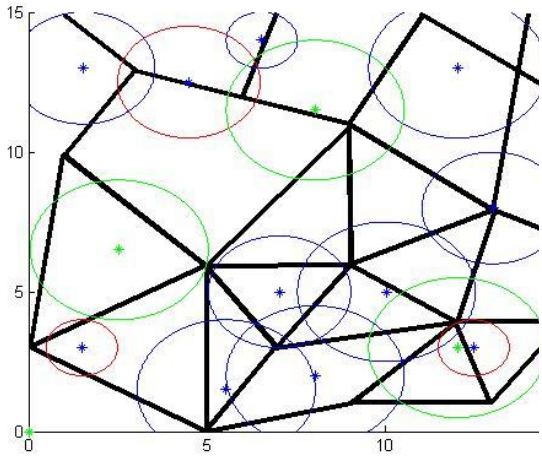

Figura 5-46

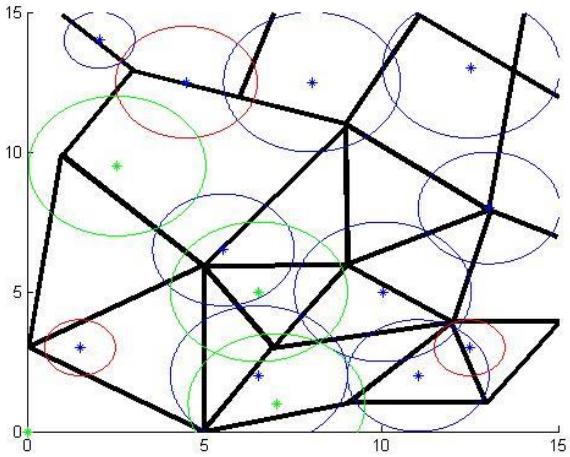

Figura 5-47

5.4.3 Caso 3

En este caso estudiaremos los resultados computacionales del código con la mejora 4 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 20, 60 y 20 respectivamente. El porcentaje de demanda requerido se ha puesto al 45%.

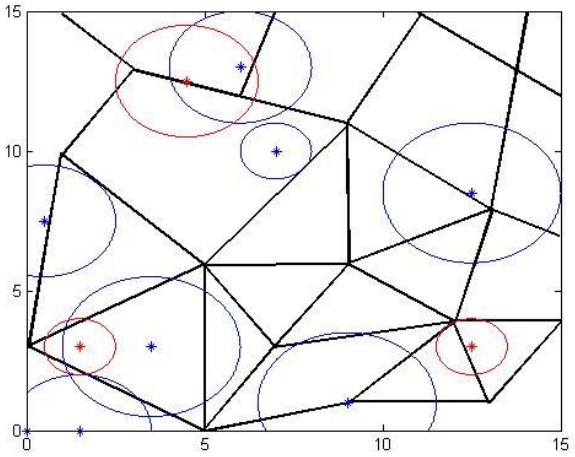

Figura 5-48

La figura 5-48 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-49 y 5-50 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas respectivamente.

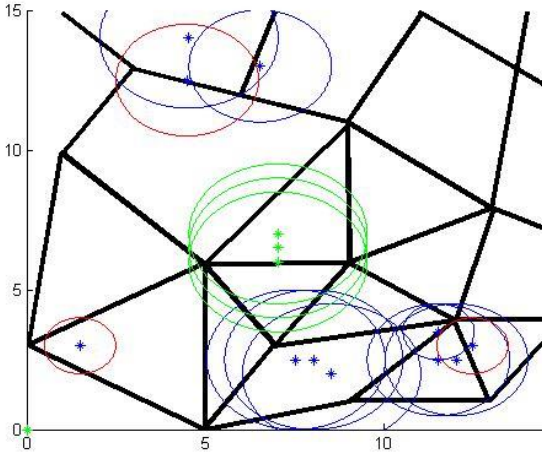

Figura 5-49

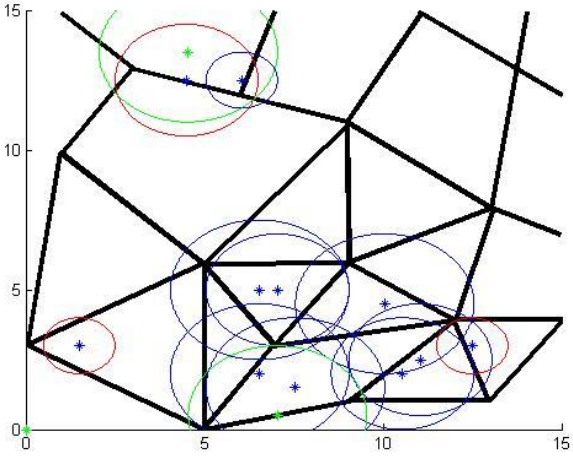

Figura 5-50

5.4.3 Caso 4

En este caso estudiaremos los resultados computacionales del código con la mejora 4 en nuestro escenario si los pesos de las coberturas simple, doble y triple son 0, 0 y 100 respectivamente. El porcentaje de demanda requerido se ha puesto al 25%.

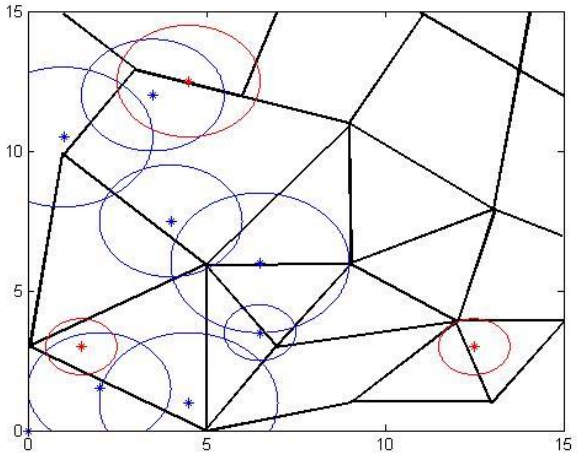

Figura 5-51

La figura 5-51 mostrada arriba muestra la solución inicial generada de forma aleatoria. Las figuras 5-52 y 5-53 muestran las soluciones finales de las soluciones iníciales aleatoria y fijas, en ambos se consiguen un porcentaje del 0.13 y 0.2484 respectivamente. En ambas soluciones no se alcanza el porcentaje exigido y se utiliza el límite máximo de estaciones extras. Sin embargo, la solución con la solución inicial fijada prácticamente cumple con lo requerido, mientras que la solución aleatoria se queda muy por debajo.

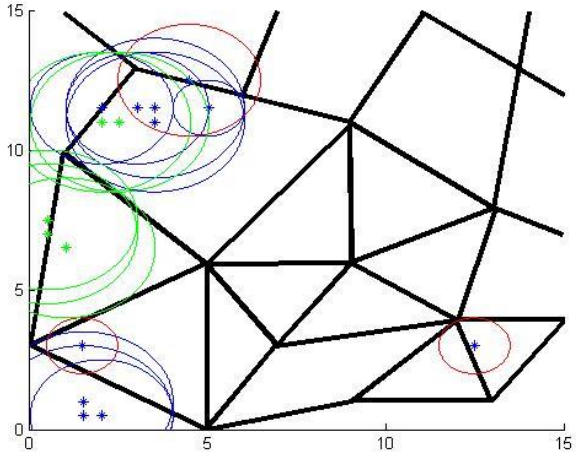

Figura 5-52

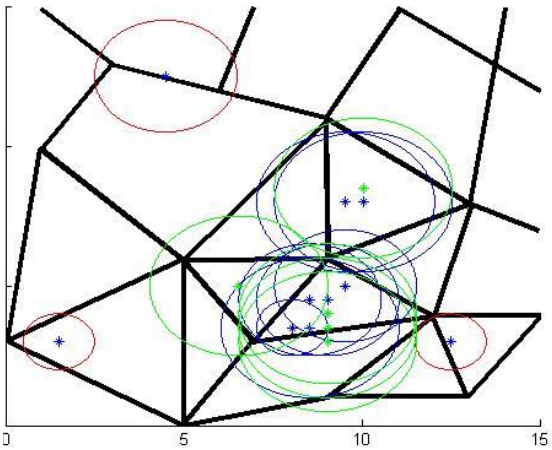

Figura 5-53

5.5 Análisis de los tiempos de ejecución

En este apartado se realizarán un conjunto de simulaciones variando en cada caso el número de puntos potenciales en el plano y el número de estaciones base a colocar. El escenario sobre el que se realizarán las simulaciones será el indicado en la Figura 5-1. En las simulaciones se usará el código base y la solución inicial se hallará de forma aleatoria.

El conjunto de simulaciones se han realizado en un ordenador con las siguientes características:

- Procesador Intel® Celeron® P4600 a 2.00GHz
- Memoria RAM 3 Gbs (2.8 utilizables)
- Windows 8 32 bits
- Versión de MATLAB: MATLAB R2013b

Número de puntos potenciales	Estaciones Base 5	Estaciones Base 15	Estaciones Base 25	Estaciones Base 35
256	19.9852 s	76.0082 s	155.9855 s	239.9757 s
441	34.2999 s	134.4941 s	261.8518 s	398.2322 s
676	56.3153 s	214.2074 s	430.4358 s	581.2104 s
961	75.6943 s	289.7171 s	591.5721 s	799.2395 s

6 CONCLUSIONES

En este capítulo expondremos algunas de las conclusiones obtenidas del análisis del algoritmo expuesto y de su análisis computacional.

En primer lugar destacar que este algoritmo está pensado para proporcionar cobertura en caminos o carreteras y no a zonas de demanda como podrían ser pueblos o ciudades, sin embargo se podría extrapolar utilizando las calles de los núcleos poblaciones y asignándole una demanda aproximada de la población cercana. Otra solución sería aplicar otro método para las zonas urbanas que considere la demanda como una zona y no como arcos.

En segundo lugar aunque nos hemos centrado en la localización de antenas de telefonía móvil, el algoritmo también podría utilizarse para satisfacer la demanda de cobertura Wifi en una zona relativamente amplia, como podría ser polígonos, grandes empresas, parques empresariales... etc. En los que se quiera conseguir una cobertura no solo dentro de los edificios sino también en los caminos, aunque en estos casos debido al precio de las infraestructuras y el uso de herramientas para medir las intensidades de las señales puede resultar un análisis trivial.

Como ya hemos comentado antes esta aproximación al problema es puramente geométrica y algunos de los problemas que ello conlleva son comentados en Berman, O.; Krass, D.; Drezner, Z (2003) [12], en donde explican que la cobertura no termina de forma abrupta sino que se va degradando con la distancia, por lo que una persona a mayor distancia del radio modelado podría disfrutar de los servicios aunque con una calidad más limitada.

En relación con lo anterior, este modelo no tiene en cuenta el entorno, el alcance o cobertura que proporciona una antena depende de los obstáculos que estén entre el teléfono y la antena, la altura de esta, las interferencias, las reflexiones...etc. Además de la posibilidad de que no se pueda instalar una antena en el lugar donde nos indique el programa debido a licencias, permiso de los propietarios del terreno o que la compensación económica al colocar una antena en ese lugar, bien a los dueños del terreno o a quienes se vean afectados por la instalación sea demasiado elevada. Por ello, puede que la solución proporcionada por el programa aunque correcta puede no ser factible.

Una solución a este problema puede ser la utilización del código de la mejora 2 dejando como fijas las estaciones que sí se pueden instalar y realizar el algoritmo de colocación con las que queremos cambiar. Otra solución podría ser añadir en una versión siguiente la posibilidad de pasarle al programa el conjunto de lugares potenciales y realizar el algoritmo sólo sobre esos puntos en vez de realizar un mallado simétrico del plano total.

En el capítulo 5 podemos apreciar las consecuencias de asignarle diferentes pesos a las cobertura simple, doble y triple, de los resultados podemos extraer que la solución inicial influye en mayor medida cuanto más alto sean

los pesos de las coberturas doble y triple. Esto se debe a que cuando le asignamos un peso muy alto a la cobertura simple el programa tiende a cubrir la mayor superficie del plano posible aunque en la solución inicial las estaciones estén más solapadas o más repartidas.

El efecto contrario se puede ver cuando se le asigna un mayor peso a las coberturas doble y triple, en estos casos el programa tiende a acercar las estaciones. El caso extremo se puede apreciar en el caso 4, en el que el algoritmo solo colocará una estación en un punto en el que su radio afecte a un segmento de un arco que ya tenga cobertura doble, pudiendo ocurrir el caso en el que una zona con una alta demanda quede sin cubrir mientras que las estaciones se agrupan en una zona con una demanda mucho más pobre, esto se debe a que en la solución inicial esta zona tuviera cobertura doble, mientras que la de mayor demanda sólo tuviera cobertura simple o no estuviera cubierta.

Un caso interesante del cálculo computacional es el que aparece en la figura 5-18. En esta imagen podemos ver cómo a pesar de que sólo se busca cobertura simple hay una instalación en la esquina inferior derecha que no aporta nada a la función objetivo, esto se debe a que la instalación con el radio más pequeño fue de las primeras estaciones en colocarse, quedando eclipsada por una estación posterior. Este problema ya fue comentado por Church and ReVelle [9] en su método GA, en el que explica que en el desarrollo del algoritmo puede que la posición de una instalación anterior no esté justificada más adelante. Este caso es el más claro, aunque puede ocurrir en los demás casos, una posible solución podría ser realizar una segunda vuelta al algoritmo, así si una estación está en su posición óptima no cambiaría de lugar, mientras que si puede mejorar la función objetivo cambiaría de posición.

En la mejora 3, cuando se empiezan a añadir las estaciones adicionales, el proceso que se sigue es equivalente al algoritmo GA en la iteración $n+k$, siendo n el número de estaciones base del algoritmo principal y k el número correspondiente de la estación adicional. Asimismo cabe resaltar la importancia de esta mejora, ya que desde el punto de vista de la operadora el objetivo principal es asegurar un determinado porcentaje de cobertura, pudiéndose realizar por ejemplo en diferentes partes, es decir, primero ejecutar el algoritmo exigiendo un porcentaje para cobertura simple, una vez encontrado el resultado, ponemos estas estaciones como fijas y exigimos un determinado porcentaje de cobertura doble y por último poner todas las estaciones de los dos resultados anteriores como fijas y exigir un determinado porcentaje de cobertura triple. Usando este método tendríamos que poner los pesos de las coberturas simple doble y triple a 100 en cada caso.

En relación con lo anterior y en consecuencia de los resultados computacionales se puede ver la importancia del límite de estaciones a añadir en la mejora 4, en especial en los casos donde la cobertura triple sea muy alta y el porcentaje exigido también sea alto. El porcentaje de demanda incrementado por la adición de cada estación base puede ser muy bajo si la solución del código principal con la cobertura triple no es muy buena, por lo que para evitar resultados inviables o saturar la ejecución se establece ese límite como control, por ejemplo en la figura 5-53 casi se cumple el porcentaje de demanda exigido habiendo cumplido ya el limite. Sin embargo, en la imagen 5-52 el porcentaje conseguido está muy por debajo del exigido

Un aspecto fundamental para abordar el problema, aunque no afecte al funcionamiento del algoritmo, es la codificación de los arcos y de las demandas asignadas a cada uno. Es fundamental un estudio inicial del escenario sobre el cual se quiere realizar el estudio si queremos llegar a una solución correcta. Seleccionar los caminos, calles, carreteras que vamos a introducir como arcos, que sean representativos atendiendo a la cantidad de demanda de servicios, proximidad...etc. Si estos caminos son curvos habrá que llevar a cabo un proceso de linealización en el cual tenemos que tener cuidado con la correcta codificación de la demanda en cada uno de los segmentos creados en la linealización. Asimismo para la asignación de la demanda a cada arco se tendrá que tener en cuenta el volumen medio de la población en el arco así como el máximo en un determinado instante del día.

A pesar de todo, atendiendo a lo explicado arriba, si tenemos una correcta codificación de los arcos y demandas y si partimos de una buena solución inicial, este algoritmo nos puede proporcionar una buena solución a nuestro problema.

7 REFERENCIAS

[1] Urueña, Alberto (coord.); Valdecasa, Elena; Ballestero, María Pilar; Ureña, Olga; Castro, Raquel; Cadenas, Santiago. Gobierno de España, Ministerio de Industria, Energía y Turismo, Secretaria de estado de telecomunicaciones y para la sociedad de la información, ONTSI. LA SOCIEDAD EN RED. Informe anual 2014.2015

[2] comScore, 2014, Distribution of times spent on social media sites in the United States in February 2014, by platform, disponible en: http://www.statista.com/statistics/294445/minutes-spent-on-us-media-sites-by-platform/

[3] Apple press info, 2015, disponible en: http://www.apple.com/pr/library/2015/01/08App-Store-Rings-in-2015-with-New-Records.html

[4] Gartner, 2014, by 2017, Mobile Users Will Provide Personalized Data Streams to More Than 100 Apps and Services Every Day, disponible en: http://www.gartner.com/newsroom/id/2654115

[5] Android; Google; App Annie; AppBrain, 2016, disponible en: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[6] Google Trends, 2016, disponible en : https://www.google.co.uk/trends/explore?date=today%203-m&q=%22pokemon%20go%22

[7] Vodafone, curiosidades y consejos, cómo se implantó la red 4g, disponible en : https://www.vodafoneteayuda.es/2013/10/como-se-implanto-la-red-4g-de-vodafone/

[8] El mundo, conectividad 5g para el 2020, disponible en : http://www.elmundo.es/tecnologia/2016/02/24/56cdb618e2704e7c2d8b456d.html

[9] Church, R.L.; ReVelle, C. The maximal covering location problem. 1974 Regional Science. 30 101-118

[10] Erdemir ET, Batta R, Spielman S, Rogerson PA, Blatt A, Flanigan M. Location coverage models with demand originating from nodes and paths: application to cellular network design. Eur J Oper Res 2008;190:610e32.

[11] Akella, M.R.; Batta, R.; Delmelle, E.M.; Rogerson, P.A.; Blatt, A.; Wilson, G. Base station location and channel allocation in a cellular network with emergency coverage requirements. 2005 European Journal of Operational Research. 164 301-323

[12] Berman, O.; Krass, D.; Drezner, Z. The gradual covering decay problem on a network. 2003 European Journal of Operational Research. 151 474-480

[13] Karasakal, O.; Karasakal, E. A maximal covering location model in the presence of partial coverage. 2004 Computers and Operations Research. 31 1515-1526

[14] Murray, Alan; Wei, Ran; Batta, Rajan. A bounding-based solution approach for the continuous arc covering problem (2014). In: Journal of Geographical Systems. RePEc:kap:jgeosy:v:16:y:2014:i:2:p:161-

182.

[15] ReVelle, C.S.; Scholssberg, M.; Williamsa, J.: Solving the maximal covering location problem with heuristic concentration. In: Computers & Operations Research 35 (2008), S. 427435

[16] Brandeau, M.L.; Chiu, S.S. An overview of representative problems in location research. 1989 Management Science. 35 645-674

[17] Cisco 2016, Global mobile data traffic from 2015 to 2020 , disponible en : http://www.statista.com/statistics/271405/global-mobile-data-traffic-forecast/

[18] Manuales, ejemplos y programa Matlab: http://es.mathworks.com/matlabcentral/index.html

ANEXOS

ANEXO A: CÓDIGO BASE

principal.m

```
function [ xBS,yBS,resultado1,porcent ] = principal(
x,y,demand,npaths,radio,nBS,single,double,triple,xBS2,yBS2,varargin)
% se puede llamar como heuristicV4(
x,y,xBS,yBS,demand,npaths,radio,nBS,single,double,triple )
% x - matriz de coordenadas origen-destino del eje x de los paths, matriz de
n_path * 2
% y - matriz de coordenadas origen-destino del eje y de los paths, matriz de
n_path * 2
% demand - vector con las demandas de los paths
% n_path - numero de paths en el sistema
% radio - radio de las BS
% nBS - numero de BS a colocar
% single - peso de cobertura simple
% double - peso de cobertura doble
% triple - peso de cobertura triple
% xBS2 - coordenadas del eje x de la solución inicial de las estaciones
% base (opcional)
% yBS2 - coordenadas del eje y de la solución inicial de las estaciones
% base (opcional)
%
%
%   Esta funcion lo que hace es recibir las coordenadas del los paths del
%   mapa, la demanda, el ancho del radio de los Bs, el numero de BS y los
%   pesos de cobertura simple, dual o triple.
%
%   Ejecuta el algoritmo heuristico, fija las posiciones de n-1 estaciones
%   base y va cambiando la posicion de la primera BS hasta hallar la
%   posicion que maximice la funcion objetivo y reitera con las siguientes
%   BS.

% argumentos: x,y,x_pot,y_pot,demand,n_paths,radio,nBS,single,double,triple


%%
```

```
%creamos la malla para las posibles posiciones de las BSs
paso=0.5;


%%
%para hayar las rectas de los paths usamos la formula de la recta y=mx+n
%pendiente de la recta que une los puntos de un segmento
pendiente=(y(:,2)-y(:,1))./(x(:,2)-x(:,1));
%punto de corte en el eje y de la recta que une los puntos de los segmentos
origenY=y(:,1)-pendiente.*x(:,1);

%como mucho una BS tendra dos puntos de cortes con cada segmento
%campos: id de la BS, id del path, 2 puntos de cortes
cortesx=zeros(npaths*nBS,4);
cortesy=zeros(npaths*nBS,4);

%variables auxiliares para guardar el óptimo temporal de la BS
cortesxdef=zeros(npaths,2);
cortesydef=zeros(npaths,2);

%llenamos las matrices con los id de las BS
for h=1:nBS
cortesx(((h-1)*npaths)+1:npaths*h,1)=h;
cortesy(((h-1)*npaths)+1:npaths*h,1)=h;
end


%% imprimir el escenario
  figure(1)
  axis([0 15 0 15]);
  hold on
  shg

for ej=1:npaths
   plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end


   %% Si no se han introducido xBS2 e yBS2 pedimos que se introduzcan
   if(nargin==9)
```

```
        disp('No has introducido los vectores de posicion de las BS, como quieres
continuar?');
        eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
        xBS=zeros(1,nBS);
        yBS=zeros(1,nBS);
        if(eleccion=='g') % se decide introducir los vectores de forma grafica
         [aux_xBS,aux_yBS]=ginput(nBS-1);
         xBS(1:(nBS-1))=aux_xBS;
         yBS(1:(nBS-1))=aux_yBS;
         malladox=0:paso:15;
         malladoy=0:paso:15;
         for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
          aux_temp=(malladox-xBS(varaux_bucle)).^2;
          indice_auxliar=(aux_temp==min(aux_temp));
          xBS(varaux_bucle)=malladox(indice_auxliar);
          aux_temp=(malladoy-yBS(varaux_bucle)).^2;
          indice_auxliar=(aux_temp==min(aux_temp));
          yBS(varaux_bucle)=malladoy(indice_auxliar);
         end


        elseif(eleccion=='a') %se decide que sea de forma aleatoria
           xBS(1:end-1)=randi([0,30],1,nBS-1)/2; %fijamos las posiciones de nBS-1
estaciones base
           yBS(1:end-1)=randi([0,30],1,nBS-1)/2;


        elseif(eleccion=='v') %se decide que sea mediante un vector
           fprintf('introtuce un vector x de %d elementos',nBS-1);
           temp_vx=input('');
           fprintf('introtuce un vector y de %d elementos \n',nBS-1);
           temp_vy=input('');
           if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
               error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
           else
             xBS(1:(nBS-1))=temp_vx;
             yBS(1:(nBS-1))=temp_vy;
             malladox=0:paso:15;
             malladoy=0:paso:15;
                 for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                     aux_temp=(malladox-xBS(varaux_bucle)).^2;
```

```
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
            end

        end



    else
        %si se han introducido xBs e yBS se comprueba que las longitudes
        %sean correctas
        if(length(xBS2)~=(nBS-1) || length(yBS2)~=(nBS-1))
            warning('los vectores yBS y xBS no tienen dimensiones correctas, como
quiere continuar?');
            eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
            xBS=zeros(1,nBS);
            yBS=zeros(1,nBS);
            if(eleccion=='g') % se decide introducir los vectores de forma grafica
              [aux_xBS,aux_yBS]=ginput(nBS-1);
              xBS(1:(nBS-1))=aux_xBS;
              yBS(1:(nBS-1))=aux_yBS;
              malladox=0:paso:15;
              malladoy=0:paso:15;
              for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                  aux_temp=(malladox-xBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  xBS(varaux_bucle)=malladox(indice_auxliar);
                  aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  yBS(varaux_bucle)=malladoy(indice_auxliar);
              end


            elseif(eleccion=='a') %se decide que sea de forma aleatoria
              xBS(1:end-1)=randi([min(min(x))*2,max(max(x))*2],1,nBS-1)/2;
%fijamos las posiciones de nBS-1 estaciones base
              yBS(1:end-1)=randi([min(min(y))*2,max(max(y))*2],1,nBS-1)/2;
```

```
        elseif(eleccion=='v') %se decide que sea mediante un vector
          fprintf('introtuce un vector x de %d elementos',nBS-1);
          temp_vx=input('');
          fprintf('introtuce un vector y de %d elementos \n',nBS-1);
          temp_vy=input('');
          if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
            error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
          else
            xBS(1:(nBS-1))=temp_vx;
            yBS(1:(nBS-1))=temp_vy;
            malladox=0:paso:15;
            malladoy=0:paso:15;
              for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                  aux_temp=(malladox-xBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  xBS(varaux_bucle)=malladox(indice_auxliar);
                  aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  yBS(varaux_bucle)=malladoy(indice_auxliar);
              end
          end


        end
      else
          xBS=[xBS2 0];
          yBS=[yBS2 0];


      end


  end


  %se imprime sobre el esscenario la solución inicial
  for ej2=1:nBS-1
    impXBS=xBS(ej2)-radio:0.001:xBS(ej2)+radio;
    impYBS1=sqrt(radio^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    impYBS2=-sqrt(radio^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    plot(impXBS,impYBS1);
    plot(impXBS,impYBS2);
  end
```

```
    plot(xBS(1:end-1),yBS(1:end-1),'*')

    hold off


%% empezamos calculando los cortes de las BS colocadas al principio
for ii=1:nBS-1
    indice=ii*npaths-npaths+1; % cada Bs tiene tantas filas reservadas para ella
como paths hay
    for jj=1:npaths % todos los paths
        %comprobamos que la estación esta dentro del rango del segmento
        %para ahorrar tiempo
        if(yBS(ii)<(max(y(jj,:))+radio) && yBS(ii)>(min(y(jj,:))-radio))
%comprobamos q esta dentro del rango del segmento

            if(xBS(ii)<(max(x(jj,:))+radio) && xBS(ii)>(min(x(jj,:))-radio))
                        %rango de valores del eje x
                        valoresXBS=xBS(ii)-radio:0.001:xBS(ii)+radio;
                        %segmento del path
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        %valores y de la circunferencia radio de la BS
                        valoresYBS1=sqrt(radio^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y buscamos cuando cambia
                        %de signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
```

```
                    cortesx(indice,4)=valoresXBS(pos(end));
                    cortesy(indice,3)=valoresYBS1(pos(1));
                    cortesy(indice,4)=valoresYBS1(pos(end));
                    aux3=2;
                end
            end

        end
        % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
        if(aux3<2)
            aux=sign(valoresYBS2-valoresy);
            aux(aux(1,:)~=1)=0;
            aux2=sum(aux);
            if(aux3==0)
                cortesx(indice,2)=jj;
                cortesy(indice,2)=jj;
            end
            if(aux2<length(aux) && aux2>0)
                pos=find(aux(1)~=aux);
                if(size(pos,2)~=0 && size(pos,2)~=1)
                    if(pos(end)==length(aux) && aux2>0)
                        if(aux3==1)
                          cortesx(indice,4)=valoresXBS(pos(1));
                          cortesy(indice,4)=valoresYBS2(pos(1));
                          aux3=2;
                        elseif (aux3==0)
                          cortesx(indice,3)=valoresXBS(pos(1));
                          cortesy(indice,3)=valoresYBS2(pos(1));
                          aux3=1;
                        end
                    else
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesx(indice,4)=valoresXBS(pos(end));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(end));
                        aux3=2;
                    end
                end

            end
        end
```

```
                        if(aux3==1)
                            %consideraciones aqui:
                            %un path siempre va a cortar al radio en dos puntos
xq
                            %esta hecho asi, por lo tanto comprobar si un punto
de
                            %corte excede los limites del path y ponerlo como el
                            %punto extremo
                            aux4=abs(xBS(ii)-x(jj,1));
                            aux5=abs(xBS(ii)-x(jj,2));
                            if(aux4==min([aux4 aux5]))
                               cortesx(indice,4)=x(jj,1);
                               cortesy(indice,4)=y(jj,1);
                            else
                               cortesx(indice,4)=x(jj,2);
                               cortesy(indice,4)=y(jj,2);
                            end
                        end

                        %resolucion de consideraciones en caso de que solo corte
en
                        %un punto
                        if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

extremo=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(extremo(1)==1)
                               cortesx(indice,3)=min(x(jj,:));
                               cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                               cortesx(indice,4)=min(x(jj,:));
                               cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

extremo=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(extremo(1)==1)
                               cortesx(indice,3)=max(x(jj,:));
                               cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                               cortesx(indice,4)=max(x(jj,:));
```

```
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                end % final de comprobacion del path




            end
          indice=indice+1;
        end
end

%calculamos la solucion inicial
resultado1=optimiza(x,y,cortesx,cortesy,npaths,demand,single,double,triple);


pause;

for ii=nBS:-1:1
    xBSdef=xBS(ii);
    yBSdef=yBS(ii);
    %la primera solución es dejarla en el mismo punto
    %esto se hace porque aunque cuando en el algoritmo vuelve a su posicion
    %no mejora nunca la solucion actual, entonces si ese es su optimo tiene
    %que quedarse ahi

    %recorremos el mallado
   for gy=0:paso:15

        for gx=0:paso:15
            %solo entra si no hay otra BS en ese punto
            if(sum(yBS(xBS==gx)==gy)==0) %solo entra si no hay otra BS en ese
punto

              yBS(ii)=gy;
              xBS(ii)=gx;
              %accedemos a la parte correspondiente de la matriz de cortes
              %ponemos la matriz de cortes correspondiente a la BS a 0
              indice=ii*npaths-npaths+1;
              cortesx(cortesx(:,1)==ii,3:4)=0;
              cortesy(cortesy(:,1)==ii,3:4)=0;
```

```
            for jj=1:npaths

                %comprobamos q esta dentro del rango del segmento
                if(yBS(ii)<(max(y(jj,:))+radio) && yBS(ii)>(min(y(jj,:))-radio))

                    if(xBS(ii)<(max(x(jj,:))+radio) && xBS(ii)>(min(x(jj,:))-
radio))
                        %creamos el rango de valores del eje x
                        valoresXBS=xBS(ii)-radio:0.001:xBS(ii)+radio;
                        %calculamos los valores y de la recta
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        % calculamos los valores y de la circunferencia de
                        % la BS
                        valoresYBS1=sqrt(radio^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y hallamos el punto
                        %donde se cambia el signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    cortesy(indice,4)=valoresYBS1(pos(end));
                                    aux3=2;
                                end
                            end
```

```
                    end
                        % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                    if(aux3<2)
                        aux=sign(valoresYBS2-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        if(aux3==0)
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                        end
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux) && aux2>0)
                                    if(aux3==1)
                                    cortesx(indice,4)=valoresXBS(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(1));
                                    aux3=2;
                                    elseif (aux3==0)
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    aux3=1;
                                    end
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(end));
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
                        %consideraciones aqui:
                        %un path siempre va a cortar al radio en dos puntos
xq
                        %esta hecho asi, por lo tanto comprobar si un punto
```

```
de
                            %corte excede los limites del path y ponerlo como el
                            %punto extremo
                            aux4=abs(xBS(ii)-x(jj,1));
                            aux5=abs(xBS(ii)-x(jj,2));
                            if(aux4==min([aux4 aux5]))
                                cortesx(indice,4)=x(jj,1);
                                cortesy(indice,4)=y(jj,1);
                            else
                                cortesx(indice,4)=x(jj,2);
                                cortesy(indice,4)=y(jj,2);
                            end
                        end

                        %resolucion de consideraciones en caso de que solo corte
en
                        %un punto
                        if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

extremo=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(extremo(1)==1)
                                cortesx(indice,3)=min(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                                cortesx(indice,4)=min(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

extremo=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(extremo(1)==1)
                                cortesx(indice,3)=max(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                    end % final de comprobacion del path
```

```
                end
              indice=indice+1;
            end

            %calculamos el resultado en ese punto

resultado2=optimiza(x,y,cortesx,cortesy,npaths,demand,single,double,triple);
            %si el resultado en este punto es mayor que el guardado,
            %sobreescribe los datos
            if(resultado2>resultado1)
                resultado1=resultado2;
                xBSdef=gx;
                yBSdef=gy;
                cortesxdef=cortesx((ii*npaths-npaths+1):(ii*npaths),3:4);
                cortesydef=cortesy((ii*npaths-npaths+1):(ii*npaths),3:4);
            end

            end

        end
    end

    %cuando llegamos al punto final guardamos la posicion del óptimo
    xBS(ii)=xBSdef;
    yBS(ii)=yBSdef;
    cortesx((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesxdef;
    cortesy((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesydef;
    cortesxdef=zeros(npaths,2);
    cortesydef=zeros(npaths,2);

end


%% imprimir el mallado
  figure(2)
  hold on
  axis([0,15,0,15]);
  shg
```

```
for ej=1:npaths
   plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',3);
end
   shg

   %imprimimos las BS y sus radios
   for ej2=1:nBS
     impXBS=xBS(ej2)-radio:0.001:xBS(ej2)+radio;
     impYBS1=sqrt(radio^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     impYBS2=-sqrt(radio^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     plot(impXBS,impYBS1);
     plot(impXBS,impYBS2);
   end
   plot(xBS(:),yBS(:),'*')
   hold off

   porcent=resultado1/sum(demand);


end
```

optimiza.m

```
function
resultado=optimiza(x,y,cortesx,cortesy,npaths,demanda,single,double,triple)
    resultado=0;
for k=1:npaths
    if((max(x(k,:))-min(x(k,:)))<0.2)
        inicio=min(y(k,:));
        final=max(y(k,:));
        t=linspace(inicio,final,10001);
        control=zeros(1,length(t));
        linea=cortesy(cortesy(:,2)==k,:);
    else
        inicio=min(x(k,:));
        final=max(x(k,:));
        t=linspace(inicio,final,10001);
        control=zeros(1,length(t));
        linea=cortesx(cortesx(:,2)==k,:);
```

```
    end

    for n=1:length(linea(:,1))
        if(linea(n,3)~=0 && linea(n,4)~=0)
            indicebajo=find(t>=min(linea(n,3:4)),1);
            indicealto=find(t>=max(linea(n,3:4)),1);
            control(indicebajo:indicealto)=control(indicebajo:indicealto)+1;
        end
    end
    % cobertura simple
    aux=sum(control>0);
    resultado=resultado+(aux/length(t))*demanda(k)*(single/100);

    %cobertura doble
    aux=sum(control>1);
    resultado=resultado+(aux/length(t))*demanda(k)*(double/100);

    %cobertura triple
    aux=sum(control>2);
    resultado=resultado+(aux/length(t))*demanda(k)*(triple/100);

end
end
```

ANEXO B: MEJORA 1 DISTINTOS RADIOS

principal.m

```
function [ xBS,yBS,resultado1,porcent ] = principal(
x,y,demand,npaths,radio,nBS,single,double,triple,xBS2,yBS2,varargin)
% se puede llamar como heuristicV4(
x,y,xBS,yBS,demand,npaths,radio,nBS,single,double,triple )
% x - matriz de coordenadas origen-destino del eje x de los paths, matriz de
n_path * 2
% y - matriz de coordenadas origen-destino del eje y de los paths, matriz de
n_path * 2
% demand - vector con las demandas de los paths
% n_path - numero de paths en el sistema
% radio - radio de las BS vector de nBS elementos
% nBS - numero de BS a colocar
% single - peso de cobertura simple
% double - peso de cobertura doble
% triple - peso de cobertura triple
% xBS2 - coordenadas del eje x de la solución inicial de las estaciones
% base (opcional)
% yBS2 - coordenadas del eje y de la solución inicial de las estaciones
% base (opcional)
%
%
%   Esta funcion lo que hace es recibir las coordenadas del los paths del
%   mapa, la demanda, el ancho del radio de los Bs, el numero de BS y los
%   pesos de cobertura simple, dual o triple.
%
%   Ejecuta el algoritmo heuristico, fija las posiciones de n-1 estaciones
%   base y va cambiando la posicion de la primera BS hasta hallar la
%   posicion que maximice la funcion objetivo y reitera con las siguientes
%   BS.

% argumentos: x,y,x_pot,y_pot,demand,n_paths,radio,nBS,single,double,triple


%%
```

```
%creamos la malla para las posibles posiciones de las BSs
paso=0.5;


%%
%para hayar las rectas de los paths usamos la formula de la recta y=mx+n
%pendiente de la recta que une los puntos de un segmento
pendiente=(y(:,2)-y(:,1))./(x(:,2)-x(:,1));
%punto de corte en el eje y de la recta que une los puntos de los segmentos
origenY=y(:,1)-pendiente.*x(:,1);

%como mucho una BS tendra dos puntos de cortes con cada segmento
%campos: id de la BS, id del path, 2 puntos de cortes
cortesx=zeros(npaths*nBS,4);
cortesy=zeros(npaths*nBS,4);

%variables auxiliares para guardar el óptimo temporal de la BS
cortesxdef=zeros(npaths,2);
cortesydef=zeros(npaths,2);

%llenamos las matrices con los id de las BS
for h=1:nBS
cortesx(((h-1)*npaths)+1:npaths*h,1)=h;
cortesy(((h-1)*npaths)+1:npaths*h,1)=h;
end


%% imprimir el escenario
  figure(1)
  axis([0 15 0 15]);
  hold on
  shg

for ej=1:npaths
   plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end


   %% Si no se han introducido xBS2 e yBS2 pedimos que se introduzcan
   if(nargin==9)
```

```
        disp('No has introducido los vectores de posicion de las BS, como quieres
continuar?');
        eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
        xBS=zeros(1,nBS);
        yBS=zeros(1,nBS);
        if(eleccion=='g') % se decide introducir los vectores de forma grafica
         [aux_xBS,aux_yBS]=ginput(nBS-1);
         xBS(1:(nBS-1))=aux_xBS;
         yBS(1:(nBS-1))=aux_yBS;
         malladox=0:paso:15;
         malladoy=0:paso:15;
         for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
          aux_temp=(malladox-xBS(varaux_bucle)).^2;
          indice_auxliar=(aux_temp==min(aux_temp));
          xBS(varaux_bucle)=malladox(indice_auxliar);
          aux_temp=(malladoy-yBS(varaux_bucle)).^2;
          indice_auxliar=(aux_temp==min(aux_temp));
          yBS(varaux_bucle)=malladoy(indice_auxliar);
         end


        elseif(eleccion=='a') %se decide que sea de forma aleatoria
           xBS(1:end-1)=randi([0,30],1,nBS-1)/2; %fijamos las posiciones de nBS-1
estaciones base
           yBS(1:end-1)=randi([0,30],1,nBS-1)/2;


        elseif(eleccion=='v') %se decide que sea mediante un vector
           fprintf('introtuce un vector x de %d elementos',nBS-1);
           temp_vx=input('');
           fprintf('introtuce un vector y de %d elementos \n',nBS-1);
           temp_vy=input('');
           if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
               error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
           else
             xBS(1:(nBS-1))=temp_vx;
             yBS(1:(nBS-1))=temp_vy;
             malladox=0:paso:15;
             malladoy=0:paso:15;
                 for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                     aux_temp=(malladox-xBS(varaux_bucle)).^2;
```

```
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
            end

        end



    else
        %si se han introducido xBs e yBS se comprueba que las longitudes
        %sean correctas
        if(length(xBS2)~=(nBS-1) || length(yBS2)~=(nBS-1))
            warning('los vectores yBS y xBS no tienen dimensiones correctas, como
quiere continuar?');
            eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
            xBS=zeros(1,nBS);
            yBS=zeros(1,nBS);
            if(eleccion=='g') % se decide introducir los vectores de forma grafica
              [aux_xBS,aux_yBS]=ginput(nBS-1);
              xBS(1:(nBS-1))=aux_xBS;
              yBS(1:(nBS-1))=aux_yBS;
              malladox=0:paso:15;
              malladoy=0:paso:15;
              for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                  aux_temp=(malladox-xBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  xBS(varaux_bucle)=malladox(indice_auxliar);
                  aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  yBS(varaux_bucle)=malladoy(indice_auxliar);
              end


            elseif(eleccion=='a') %se decide que sea de forma aleatoria
              xBS(1:end-1)=randi([min(min(x))*2,max(max(x))*2],1,nBS-1)/2;
%fijamos las posiciones de nBS-1 estaciones base
              yBS(1:end-1)=randi([min(min(y))*2,max(max(y))*2],1,nBS-1)/2;
```

```
        elseif(eleccion=='v') %se decide que sea mediante un vector
          fprintf('introtuce un vector x de %d elementos',nBS-1);
          temp_vx=input('');
          fprintf('introtuce un vector y de %d elementos \n',nBS-1);
          temp_vy=input('');
          if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
            error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
          else
            xBS(1:(nBS-1))=temp_vx;
            yBS(1:(nBS-1))=temp_vy;
            malladox=0:paso:15;
            malladoy=0:paso:15;
              for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                  aux_temp=(malladox-xBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  xBS(varaux_bucle)=malladox(indice_auxliar);
                  aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  yBS(varaux_bucle)=malladoy(indice_auxliar);
              end
          end


        end
      else
          xBS=[xBS2 0];
          yBS=[yBS2 0];


      end


  end

  %se imprime sobre el esscenario la solución inicial
  for ej2=1:nBS-1
    impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
    impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    plot(impXBS,impYBS1);
    plot(impXBS,impYBS2);
  end
```

```
   plot(xBS(1:end-1),yBS(1:end-1),'*')

   hold off


%% empezamos calculando los cortes de las BS colocadas al principio
for ii=1:nBS-1 % todos los BS -1 fijos
    indice=ii*npaths-npaths+1; % cada Bs tiene tantas filas reservadas para ella
como paths hay
    for jj=1:npaths
      %comprobamos que la estación esta dentro del rango del segmento
      %para ahorrar tiempo
        if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-radio(ii)))
%comprobamos q esta dentro del rango del segmento


            if(xBS(ii)<(max(x(jj,:))+radio(ii)) && xBS(ii)>(min(x(jj,:))-
radio(ii)))
                        %rango de valores del eje x
                        valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
                        %segmento del path
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y buscamos cuando cambia
                        %de signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
```

```
                    cortesy(indice,3)=valoresYBS1(pos(1));
                    cortesy(indice,4)=valoresYBS1(pos(end));
                    aux3=2;
                end
            end

        end
                % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
        if(aux3<2)
            aux=sign(valoresYBS2-valoresy);
            aux(aux(1,:)~=1)=0;
            aux2=sum(aux);
            if(aux3==0)
                cortesx(indice,2)=jj;
                cortesy(indice,2)=jj;
            end
            if(aux2<length(aux) && aux2>0)
                pos=find(aux(1)~=aux);
                if(size(pos,2)~=0 && size(pos,2)~=1)
                    if(pos(end)==length(aux) && aux2>0)
                        if(aux3==1)
                        %consideraciones aqui:
                        %un path siempre va a cortar al radio en
dos puntos xq
                        %esta hecho asi, por lo tanto comprobar
si un punto de
                        %corte excede los limites del path y
ponerlo como el
                        %punto extremo
                        cortesx(indice,4)=valoresXBS(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(1));
                        aux3=2;
                        elseif (aux3==0)
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        aux3=1;
                        end
                    else
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesx(indice,4)=valoresXBS(pos(end));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(end));
```

```
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
                        aux4=abs(xBS(ii)-x(jj,1));
                        aux5=abs(xBS(ii)-x(jj,2));
                        if(aux4==min([aux4 aux5]))
                            cortesx(indice,4)=x(jj,1);
                            cortesy(indice,4)=y(jj,1);
                        else
                            cortesx(indice,4)=x(jj,2);
                            cortesy(indice,4)=y(jj,2);
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=min(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                        else
                            cortesx(indice,4)=min(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                        end
                    end

                    if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=max(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                        else
                            cortesx(indice,4)=max(x(jj,:));
```

```
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                end % final de comprobacion del path

            end
          indice=indice+1;
        end
end

%calculamos la solucion inicial
resultado1=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);

pause;

for ii=nBS:-1:1
    xBSdef=xBS(ii);
    yBSdef=yBS(ii);
    %la primera solución es dejarla en el mismo punto
    %esto se hace porque aunque cuando en el algoritmo vuelve a su posicion
    %no mejora nunca la solucion actual, entonces si ese es su optimo tiene
    %que quedarse ahi
   for gy=min(min(y)):paso:max(max(y))
       %recorremos el mallado
       for gx=min(min(x)):paso:max(max(x))

            %solo entra si no hay otra BS en ese punto
            if(sum(yBS(xBS==gx)==gy)==0)
            yBS(ii)=gy;
            xBS(ii)=gx;

            %accedemos a la parte correspondiente de la matriz de cortes
            %ponemos la matriz de cortes correspondiente a la BS a 0
            indice=ii*npaths-npaths+1;
            cortesx(cortesx(:,1)==ii,3:4)=0;
            cortesy(cortesy(:,1)==ii,3:4)=0;

            for jj=1:npaths % todos los paths

                %comprobamos q esta dentro del rango del segmento
                if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-
radio(ii)))
```

```
                    if(xBS(ii)<(max(x(jj,:))+radio(ii)) &&
xBS(ii)>(min(x(jj,:))-radio(ii)))
                        %creamos el rango de valores del eje x
                        valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
                        %calculamos los valores y de la recta
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y hallamos el punto
                        %donde se cambia el signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    cortesy(indice,4)=valoresYBS1(pos(end));
                                    aux3=2;
                                end
                             end

                        end
                            % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                        if(aux3<2)
                            aux=sign(valoresYBS2-valoresy);
                            aux(aux(1,:)~=1)=0;
                            aux2=sum(aux);
```

```
                    if(aux3==0)
                        cortesx(indice,2)=jj;
                        cortesy(indice,2)=jj;
                    end
                    if(aux2<length(aux) && aux2>0)
                        pos=find(aux(1)~=aux);
                        if(size(pos,2)~=0 && size(pos,2)~=1)
                            if(pos(end)==length(aux) && aux2>0)
                                if(aux3==1)
                                %consideraciones aqui:
                                %un path siempre va a cortar al radio en
dos puntos xq
                                %esta hecho asi, por lo tanto comprobar
si un punto de
                                %corte excede los limites del path y
ponerlo como el
                                %punto extremo
                                cortesx(indice,4)=valoresXBS(pos(1));
                                cortesy(indice,4)=valoresYBS2(pos(1));
                                aux3=2;
                                elseif (aux3==0)
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesy(indice,3)=valoresYBS2(pos(1));
                                aux3=1;
                                end
                            else
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesx(indice,4)=valoresXBS(pos(end));
                                cortesy(indice,3)=valoresYBS2(pos(1));
                                cortesy(indice,4)=valoresYBS2(pos(end));
                                aux3=2;
                            end
                        end

                    end
                end

                if(aux3==1)
                    aux4=abs(xBS(ii)-x(jj,1));
                    aux5=abs(xBS(ii)-x(jj,2));
                    if(aux4==min([aux4 aux5]))
                       cortesx(indice,4)=x(jj,1);
```

```
                                cortesy(indice,4)=y(jj,1);
                            else
                                cortesx(indice,4)=x(jj,2);
                                cortesy(indice,4)=y(jj,2);
                            end
                        end

                        %resolucion de consideraciones en caso de que solo corte
en
                        %un punto
                        if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=min(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                                cortesx(indice,4)=min(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=max(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                    end % final de comprobacion del path


                end
              indice=indice+1;
            end

            %calculamos el resultado en ese punto
```

```
resultado2=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);
            %si el resultado en este punto es mayor que el guardado,
            %sobreescribe los datos
            if(resultado2>resultado1)
                resultado1=resultado2;
                xBSdef=gx;
                yBSdef=gy;
                cortesxdef=cortesx((ii*npaths-npaths+1):(ii*npaths),3:4);
                cortesydef=cortesy((ii*npaths-npaths+1):(ii*npaths),3:4);
            end

            end


        end
    end
    %cuando llegamos al punto final guardamos la posicion del óptimo
    xBS(ii)=xBSdef;
    yBS(ii)=yBSdef;
    cortesx((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesxdef;
    cortesy((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesydef;
    cortesxdef=zeros(npaths,2);
    cortesydef=zeros(npaths,2);


end

%% imprimir el mallado
  figure(2)
    hold on
    axis([0,15,0,15]);

    shg
for ej=1:npaths
    plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end
    shg

    %% imprimimos la posicion de las BS y sus radios

    for ej2=1:nBS
```

```
    impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
    impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
    plot(impXBS,impYBS1);
    plot(impXBS,impYBS2);
    end
    plot(xBS,yBS,'*');
    hold off

    porcent=resultado1/sum(demand);

end
```

ANEXO C: MEJORA 2 ESTACIONES BASE FIJAS

principal.m

```
function [ xBS,yBS,resultado1,porcent ] = principal(
x,y,demand,npaths,radio,nBS,single,double,triple,xBSfijos,yBSfijos,radiofijos,xB
S2,yBS2,varargin)
% se puede llamar como heuristicV4(
x,y,xBS,yBS,demand,npaths,radio,nBS,single,double,triple )
% x - matriz de coordenadas origen-destino del eje x de los paths, matriz de
n_path * 2
% y - matriz de coordenadas origen-destino del eje y de los paths, matriz de
n_path * 2
% demand - vector con las demandas de los paths
% n_path - numero de paths en el sistema
% radio - radio de las BS vector de nBS elementos
% nBS - numero de BS a colocar
% single - peso de cobertura simple
% double - peso de cobertura doble
% triple - peso de cobertura triple
% xBSfijos - coordenadas del eje x de las estaciones base fijas
% yBSfijos - coordenadas del eje y de las estaciones base fijas
% xBS2 - coordenadas del eje x de la solución inicial de las estaciones
% base (opcional)
% yBS2 - coordenadas del eje y de la solución inicial de las estaciones
% base (opcional)
%
%
%   Esta funcion lo que hace es recibir las coordenadas del los paths del
%   mapa, la demanda, el ancho del radio de los Bs, el numero de BS y los
%   pesos de cobertura simple, dual o triple.
%
%   Ejecuta el algoritmo heuristico, fija las posiciones de n-1 estaciones
%   base y va cambiando la posicion de la primera BS hasta hallar la
%   posicion que maximice la funcion objetivo y reitera con las siguientes
%   BS.

% argumentos: x,y,x_pot,y_pot,demand,n_paths,radio,nBS,single,double,triple
```

```
%%


%creamos la malla para las posibles posiciones de las BSs
paso=0.5;

%para hayar las rectas de los paths usamos la formula de la recta y=mx+n
%pendiente de la recta que une los puntos de un segmento
pendiente=(y(:,2)-y(:,1))./(x(:,2)-x(:,1));
%punto de corte en el eje y de la recta que une los puntos de los segmentos
origenY=y(:,1)-pendiente.*x(:,1);

%como mucho una BS tendra dos puntos de cortes con cada segmento
%campos: id de la BS, id del path, 2 puntos de cortes
cortesx=zeros(npaths*nBS,4);
cortesy=zeros(npaths*nBS,4);

%variables auxiliares para guardar el óptimo temporal de la BS
cortesxdef=zeros(npaths,2);
cortesydef=zeros(npaths,2);

%llenamos las matrices con los id de las BS
for h=1:nBS
cortesx(((h-1)*npaths)+1:npaths*h,1)=h;
cortesy(((h-1)*npaths)+1:npaths*h,1)=h;
end


%% imprimir el mallado
  figure(1)
  axis([0 15 0 15]);
  hold on
  shg

for ej=1:npaths
   plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end


   %% imprimimos la posicion de las BS
```

```
   if(nargin==12)
       disp('No has introducido los vectores de posicion de las BS, como quieres
continuar?');
       eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
       xBS=zeros(1,nBS);
       yBS=zeros(1,nBS);
       if(eleccion=='g') % se decide introducir los vectores de forma grafica
        [aux_xBS,aux_yBS]=ginput(nBS-1);
        xBS(1:(nBS-1))=aux_xBS;
        yBS(1:(nBS-1))=aux_yBS;
        malladox=0:paso:15;
        malladoy=0:paso:15;
        for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
         aux_temp=(malladox-xBS(varaux_bucle)).^2;
         indice_auxliar=(aux_temp==min(aux_temp));
         xBS(varaux_bucle)=malladox(indice_auxliar);
         aux_temp=(malladoy-yBS(varaux_bucle)).^2;
         indice_auxliar=(aux_temp==min(aux_temp));
         yBS(varaux_bucle)=malladoy(indice_auxliar);
        end

       elseif(eleccion=='a') %se decide que sea de forma aleatoria
          xBS(1:end-1)=randi([0,30],1,nBS-1)/2; %fijamos las posiciones de nBS-1
estaciones base
          yBS(1:end-1)=randi([0,30],1,nBS-1)/2;

       elseif(eleccion=='v') %se decide que sea mediante un vector
          fprintf('introtuce un vector x de %d elementos',nBS-1);
          temp_vx=input('');
          fprintf('introtuce un vector y de %d elementos \n',nBS-1);
          temp_vy=input('');
          if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
              error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
          else
            xBS(1:(nBS-1))=temp_vx;
            yBS(1:(nBS-1))=temp_vy;
            malladox=0:paso:15;
            malladoy=0:paso:15;
                for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
```

```
                    aux_temp=(malladox-xBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
            end

        end



    else
        %si se han introducido xBs e yBS se comprueba que las longitudes
        %sean correctas
        if(length(xBS2)~=(nBS-1) || length(yBS2)~=(nBS-1))
            warning('los vectores yBS y xBS no tienen dimensiones correctas, como
quiere continuar?');
            eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
            xBS=zeros(1,nBS);
            yBS=zeros(1,nBS);
            if(eleccion=='g') % se decide introducir los vectores de forma grafica
              [aux_xBS,aux_yBS]=ginput(nBS-1);
              xBS(1:(nBS-1))=aux_xBS;
              yBS(1:(nBS-1))=aux_yBS;
              malladox=0:paso:15;
              malladoy=0:paso:15;
              for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                  aux_temp=(malladox-xBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  xBS(varaux_bucle)=malladox(indice_auxliar);
                  aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                  indice_auxliar=(aux_temp==min(aux_temp));
                  yBS(varaux_bucle)=malladoy(indice_auxliar);
              end


            elseif(eleccion=='a') %se decide que sea de forma aleatoria
              xBS(1:end-1)=randi([min(min(x))*2,max(max(x))*2],1,nBS-1)/2;
%fijamos las posiciones de nBS-1 estaciones base
```

```
            yBS(1:end-1)=randi([min(min(y))*2,max(max(y))*2],1,nBS-1)/2;

          elseif(eleccion=='v') %se decide que sea mediante un vector
            fprintf('introtuce un vector x de %d elementos',nBS-1);
            temp_vx=input('');
            fprintf('introtuce un vector y de %d elementos \n',nBS-1);
            temp_vy=input('');
            if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
              error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
            else
              xBS(1:(nBS-1))=temp_vx;
              yBS(1:(nBS-1))=temp_vy;
              malladox=0:paso:15;
              malladoy=0:paso:15;
                for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                    aux_temp=(malladox-xBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
            end

          end
        else
            xBS=[xBS2 0];
            yBS=[yBS2 0];

        end

    end

    %se imprime sobre el escenario la solución inicial
    for ej2=1:nBS-1
      impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
      impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
      impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
      plot(impXBS,impYBS1);
      plot(impXBS,impYBS2);
```

```
    end
    plot(xBS(1:end-1),yBS(1:end-1),'*')
    for ej2=1:length(xBSfijos)
     impXBS=xBSfijos(ej2)-radiofijos(ej2):0.001:xBSfijos(ej2)+radiofijos(ej2);
     impYBS1=sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     impYBS2=-sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     plot(impXBS,impYBS1,'r');
     plot(impXBS,impYBS2,'r');
    end
    plot(xBSfijos,yBSfijos,'r*');

    hold off


%% empezamos calculando los cortes de las BS colocadas al principio
for ii=1:nBS-1 % todos los BS -1 fijos
    indice=ii*npaths-npaths+1; % cada Bs tiene tantas filas reservadas para ella
como paths hay
    for jj=1:npaths
      %comprobamos que la estación esta dentro del rango del segmento
      %para ahorrar tiempo
        if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-radio(ii)))
%comprobamos q esta dentro del rango del segmento

            if(xBS(ii)<(max(x(jj,:))+radio(ii)) && xBS(ii)>(min(x(jj,:))-
radio(ii)))
                        %rango de valores del eje x
                        valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
                        %segmento del path
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y buscamos cuando cambia
                        %de signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
```

```
                    cortesy(indice,2)=jj;
                    if(size(pos,2)~=0 && size(pos,2)~=1)
                        if(pos(end)==length(aux))
                            cortesx(indice,3)=valoresXBS(pos(1));
                            cortesy(indice,3)=valoresYBS1(pos(1));
                            aux3=1;
                        else
                            cortesx(indice,3)=valoresXBS(pos(1));
                            cortesx(indice,4)=valoresXBS(pos(end));
                            cortesy(indice,3)=valoresYBS1(pos(1));
                            cortesy(indice,4)=valoresYBS1(pos(end));
                            aux3=2;
                        end
                     end

                end
                    % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                if(aux3<2)
                    aux=sign(valoresYBS2-valoresy);
                    aux(aux(1,:)~=1)=0;
                    aux2=sum(aux);
                    if(aux3==0)
                        cortesx(indice,2)=jj;
                        cortesy(indice,2)=jj;
                    end
                    if(aux2<length(aux) && aux2>0)
                        pos=find(aux(1)~=aux);
                        if(size(pos,2)~=0 && size(pos,2)~=1)
                            if(pos(end)==length(aux) && aux2>0)
                                if(aux3==1)
                                %consideraciones aqui:
                                %un path siempre va a cortar al radio en
dos puntos xq
                                %esta hecho asi, por lo tanto comprobar
si un punto de
                                %corte excede los limites del path y
ponerlo como el
                                %punto extremo
                                cortesx(indice,4)=valoresXBS(pos(1));
                                cortesy(indice,4)=valoresYBS2(pos(1));
                                aux3=2;
                                elseif (aux3==0)
```

```
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    aux3=1;
                                    end
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(end));
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
                        aux4=abs(xBS(ii)-x(jj,1));
                        aux5=abs(xBS(ii)-x(jj,2));
                        if(aux4==min([aux4 aux5]))
                            cortesx(indice,4)=x(jj,1);
                            cortesy(indice,4)=y(jj,1);
                        else
                            cortesx(indice,4)=x(jj,2);
                            cortesy(indice,4)=y(jj,2);
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=min(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                        else
                            cortesx(indice,4)=min(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                        end
                    end
```

```
                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=max(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
            end % final de comprobacion del path

        end
      indice=indice+1;
    end
end

%hasta aqui los normales ahora los fijos

%llenamos las matrices con los id de las BS
for h=1:length(xBSfijos)
cortesx(nBS*npaths+1+(h-1)*npaths:nBS*npaths+npaths*h,1)=-1*h;
cortesy(nBS*npaths+1+(h-1)*npaths:nBS*npaths+npaths*h,1)=-1*h;
end

for ii=1:length(xBSfijos) % todos los BS -1 fijos
    indice=nBS*npaths+ii*npaths-npaths+1; % cada Bs tiene tantas filas
reservadas para ella como paths hay
    for jj=1:npaths % todos los paths
        %comprobamos que la estación esta dentro del rango del segmento
        %para ahorrar tiempo
        if(yBSfijos(ii)<(max(y(jj,:))+radiofijos(ii)) &&
yBSfijos(ii)>(min(y(jj,:))-radiofijos(ii))) %comprobamos q esta dentro del rango
del segmento

            if(xBSfijos(ii)<(max(x(jj,:))+radiofijos(ii)) &&
xBSfijos(ii)>(min(x(jj,:))-radiofijos(ii)))
                        %rango de valores del eje x
                        valoresXBS=xBSfijos(ii)-
radiofijos(ii):0.001:xBSfijos(ii)+radiofijos(ii);
                        %segmento del path
```

```
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        %valores y de la circunferencia radio de la BS
                        valoresYBS1=sqrt(radiofijos(ii)^2-(valoresXBS-
xBSfijos(ii)).^2)+yBSfijos(ii);
                        valoresYBS2=-sqrt(radiofijos(ii)^2-(valoresXBS-
xBSfijos(ii)).^2)+yBSfijos(ii);
                        %restamos los valores de y y buscamos cuando cambia
                        %de signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    cortesy(indice,4)=valoresYBS1(pos(end));
                                    aux3=2;
                                end
                             end

                        end
                                % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                        if(aux3<2)
                            aux=sign(valoresYBS2-valoresy);
                            aux(aux(1,:)~=1)=0;
                            aux2=sum(aux);
                            if(aux3==0)
                                cortesx(indice,2)=jj;
                                cortesy(indice,2)=jj;
                            end
                            if(aux2<length(aux) && aux2>0)
```

```
                pos=find(aux(1)~=aux);
                if(size(pos,2)~=0 && size(pos,2)~=1)
                    if(pos(end)==length(aux) && aux2>0)
                        if(aux3==1)
                        %consideraciones aqui:
                        %un path siempre va a cortar al radio en
dos puntos xq
                        %esta hecho asi, por lo tanto comprobar
si un punto de
                        %corte excede los limites del path y
ponerlo como el
                        %punto extremo
                        cortesx(indice,4)=valoresXBS(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(1));
                        aux3=2;
                        elseif (aux3==0)
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        aux3=1;
                        end
                    else
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesx(indice,4)=valoresXBS(pos(end));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(end));
                        aux3=2;
                    end
                end

            end
        end

        if(aux3==1)
            aux4=abs(xBSfijos(ii)-x(jj,1));
            aux5=abs(xBSfijos(ii)-x(jj,2));
            if(aux4==min([aux4 aux5]))
               cortesx(indice,4)=x(jj,1);
               cortesy(indice,4)=y(jj,1);
            else
               cortesx(indice,4)=x(jj,2);
               cortesy(indice,4)=y(jj,2);
            end
```

```
                        end

                        %resolucion de consideraciones en caso de que solo corte
en
                        %un punto
                        if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(socorro(1)==1)
                               cortesx(indice,3)=min(x(jj,:));
                               cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                               cortesx(indice,4)=min(x(jj,:));
                               cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(socorro(1)==1)
                               cortesx(indice,3)=max(x(jj,:));
                               cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                               cortesx(indice,4)=max(x(jj,:));
                               cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                end % final de comprobacion del path



            end
          indice=indice+1;
        end
end

% imprimimos la primera solucion
resultado1=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);

%% algoritmo principal
```

```
pause;


for ii=nBS:-1:1
    xBSdef=xBS(ii);
    yBSdef=yBS(ii);
    %la primera solución es dejarla en el mismo punto
    %esto se hace porque aunque cuando en el algoritmo vuelve a su posicion
    %no mejora nunca la solucion actual, entonces si ese es su optimo tiene
    %que quedarse ahi
   for gy=min(min(y)):paso:max(max(y))
       %recorremos el mallado
       for gx=min(min(x)):paso:max(max(x))

            %solo entra si no hay otra BS en ese punto
            if(sum(yBS(xBS==gx)==gy)==0) %solo entra si no hay otra BS en ese
punto
             yBS(ii)=gy;
             xBS(ii)=gx;
             %accedemos a la parte correspondiente de la matriz de cortes
             %ponemos la matriz de cortes correspondiente a la BS a 0
             indice=ii*npaths-npaths+1;
             cortesx(cortesx(:,1)==ii,3:4)=0;
             cortesy(cortesy(:,1)==ii,3:4)=0;

            for jj=1:npaths % todos los paths

                %comprobamos q esta dentro del rango del segmento
                if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-
radio(ii)))

                    if(xBS(ii)<(max(x(jj,:))+radio(ii)) &&
xBS(ii)>(min(x(jj,:))-radio(ii)))
                        %creamos el rango de valores del eje x
                        valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
                        %calculamos los valores y de la recta
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y hallamos el punto
                        %donde se cambia el signo
                        aux=sign(valoresYBS1-valoresy);
```

```
                    aux(aux(1,:)~=1)=0;
                    aux2=sum(aux);
                    aux3=0;
                    if(aux2<length(aux) && aux2>0)
                        pos=find(aux(1)~=aux);
                        cortesx(indice,2)=jj;
                        cortesy(indice,2)=jj;
                        if(size(pos,2)~=0 && size(pos,2)~=1)
                            if(pos(end)==length(aux))
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesy(indice,3)=valoresYBS1(pos(1));
                                aux3=1;
                            else
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesx(indice,4)=valoresXBS(pos(end));
                                cortesy(indice,3)=valoresYBS1(pos(1));
                                cortesy(indice,4)=valoresYBS1(pos(end));
                                aux3=2;
                            end
                         end

                    end
                            % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                    if(aux3<2)
                        aux=sign(valoresYBS2-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        if(aux3==0)
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                        end
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux) && aux2>0)
                                    if(aux3==1)
                                    %consideraciones aqui:
                                    %un path siempre va a cortar al radio en
dos puntos xq
                                    %esta hecho asi, por lo tanto comprobar
si un punto de
```

```
                                        %corte excede los limites del path y
ponerlo como el
                                        %punto extremo
                                        cortesx(indice,4)=valoresXBS(pos(1));
                                        cortesy(indice,4)=valoresYBS2(pos(1));
                                        aux3=2;
                                        elseif (aux3==0)
                                        cortesx(indice,3)=valoresXBS(pos(1));
                                        cortesy(indice,3)=valoresYBS2(pos(1));
                                        aux3=1;
                                        end
                                    else
                                        cortesx(indice,3)=valoresXBS(pos(1));
                                        cortesx(indice,4)=valoresXBS(pos(end));
                                        cortesy(indice,3)=valoresYBS2(pos(1));
                                        cortesy(indice,4)=valoresYBS2(pos(end));
                                        aux3=2;
                                    end
                                end

                            end
                        end

                        if(aux3==1)
                            aux4=abs(xBS(ii)-x(jj,1));
                            aux5=abs(xBS(ii)-x(jj,2));
                            if(aux4==min([aux4 aux5]))
                               cortesx(indice,4)=x(jj,1);
                               cortesy(indice,4)=y(jj,1);
                            else
                               cortesx(indice,4)=x(jj,2);
                               cortesy(indice,4)=y(jj,2);
                            end
                        end

                        %resolucion de consideraciones en caso de que solo corte
en
                        %un punto
                        if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(socorro(1)==1)
```

```
                                cortesx(indice,3)=min(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                                cortesx(indice,4)=min(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=max(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                    end % final de comprobacion del path

                end
              indice=indice+1;
            end

            %calculamos el resultado

resultado2=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);
            %si el resultado en este punto es mayor que el guardado,
            %sobreescribe los datos
            if(resultado2>resultado1)
                resultado1=resultado2;
                xBSdef=gx;
                yBSdef=gy;
                cortesxdef=cortesx((ii*npaths-npaths+1):(ii*npaths),3:4);
                cortesydef=cortesy((ii*npaths-npaths+1):(ii*npaths),3:4);
            end

            end


        end
```

```
    end
    %cuando llegamos al punto final guardamos la posicion del óptimo
    xBS(ii)=xBSdef;
    yBS(ii)=yBSdef;
    cortesx((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesxdef;
    cortesy((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesydef;
    cortesxdef=zeros(npaths,2);
    cortesydef=zeros(npaths,2);

end
%% imprimir el mallado
  figure(2)
    hold on
    axis([0,15,0,15]);

    shg
for ej=1:npaths
    plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end
    %% imprimimos la posicion de las BS
    for ej2=1:nBS
     impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
     impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     plot(impXBS,impYBS1);
     plot(impXBS,impYBS2);
    end
    plot(xBS,yBS,'*');
    %imprimimos las fijas
    for ej2=1:length(xBSfijos)
     impXBS=xBSfijos(ej2)-radiofijos(ej2):0.001:xBSfijos(ej2)+radiofijos(ej2);
     impYBS1=sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     impYBS2=-sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     plot(impXBS,impYBS1,'r');
     plot(impXBS,impYBS2,'r');
    end
    plot(xBSfijos,yBSfijos,'r*');
    hold off

    porcent=resultado1/sum(demand);
end
```

ANEXO D: MEJORA 3 PORCENTAJE DE DEMANDA

principal.m

```
function [ xBS,yBS,resultado1,porcent ] = principal (
x,y,demand,npaths,radio,nBS,single,double,triple,xBSfijos,yBSfijos,radiofijos,po
rcentaje,exceso,xBS2,yBS2,varargin)
% se puede llamar como heuristicV4(
x,y,xBS,yBS,demand,npaths,radio,nBS,single,double,triple )
% x - matriz de coordenadas origen-destino del eje x de los paths, matriz de
n_path * 2
% y - matriz de coordenadas origen-destino del eje y de los paths, matriz de
n_path * 2
% demand - vector con las demandas de los paths
% n_path - numero de paths en el sistema
% radio - radio de las BS vector de nBS elementos
% nBS - numero de BS a colocar
% single - peso de cobertura simple
% double - peso de cobertura doble
% triple - peso de cobertura triple
% xBSfijos - coordenadas del eje x de las estaciones base fijas
% yBSfijos - coordenadas del eje y de las estaciones base fijas
% porcentaje - porcentaje de demanda minimo a cumplir
% exceso - variable de control para indicar el número máximo de estaciones
% adicionales a colocar
% xBS2 - coordenadas del eje x de la solución inicial de las estaciones
% base (opcional)
% yBS2 - coordenadas del eje y de la solución inicial de las estaciones
% base (opcional)
%
%
%   Esta funcion lo que hace es recibir las coordenadas del los paths del
%   mapa, la demanda, el ancho del radio de los Bs, el numero de BS y los
%   pesos de cobertura simple, dual o triple.
%
```

```
%   Ejecuta el algoritmo heuristico, fija las posiciones de n-1 estaciones
%   base y va cambiando la posicion de la primera BS hasta hallar la
%   posicion que maximice la funcion objetivo y reitera con las siguientes
%   BS.

% argumentos: x,y,x_pot,y_pot,demand,n_paths,radio,nBS,single,double,triple


%%

%creamos la malla para las posibles posiciones de las BSs
paso=0.5;

%para hayar las rectas de los paths usamos la formula de la recta y=mx+n
%pendiente de la recta que une los puntos de un segmento
pendiente=(y(:,2)-y(:,1))./(x(:,2)-x(:,1));
%punto de corte en el eje y de la recta que une los puntos de los segmentos
origenY=y(:,1)-pendiente.*x(:,1);

%como mucho una BS tendra dos puntos de cortes con cada segmento
%campos: id de la BS, id del path, 2 puntos de cortes
cortesx=zeros(npaths*nBS,4);
cortesy=zeros(npaths*nBS,4);

%variables auxiliares para guardar el óptimo temporal de la BS
cortesxdef=zeros(npaths,2);
cortesydef=zeros(npaths,2);

%llenamos las matrices con los id de las BS
for h=1:nBS
cortesx(((h-1)*npaths)+1:npaths*h,1)=h;
cortesy(((h-1)*npaths)+1:npaths*h,1)=h;
end


%% imprimir el mallado
%% imprimir el mallado
  figure(1)
  axis([0 15 0 15]);
  hold on
  shg
```

```
for ej=1:npaths
   plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',2);
end


  shg


   %% imprimimos la posicion de las BS
   if(nargin==14)
       disp('No has introducido los vectores de posicion de las BS, como quieres
continuar?');
       eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
       xBS=zeros(1,nBS);
       yBS=zeros(1,nBS);
       if(eleccion=='g') % se decide introducir los vectores de forma grafica
        [aux_xBS,aux_yBS]=ginput(nBS-1);
        xBS(1:(nBS-1))=aux_xBS;
        yBS(1:(nBS-1))=aux_yBS;
        malladox=0:paso:15;
        malladoy=0:paso:15;
        for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
         aux_temp=(malladox-xBS(varaux_bucle)).^2;
         indice_auxliar=(aux_temp==min(aux_temp));
         xBS(varaux_bucle)=malladox(indice_auxliar);
         aux_temp=(malladoy-yBS(varaux_bucle)).^2;
         indice_auxliar=(aux_temp==min(aux_temp));
         yBS(varaux_bucle)=malladoy(indice_auxliar);
        end


       elseif(eleccion=='a') %se decide que sea de forma aleatoria
          xBS(1:end-1)=randi([0,30],1,nBS-1)/2; %fijamos las posiciones de nBS-1
estaciones base
          yBS(1:end-1)=randi([0,30],1,nBS-1)/2;


       elseif(eleccion=='v') %se decide que sea mediante un vector
          fprintf('introtuce un vector x de %d elementos',nBS-1);
          temp_vx=input('');
          fprintf('introtuce un vector y de %d elementos \n',nBS-1);
          temp_vy=input('');
          if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
              error('alguno de los vectores introducidos no tiene las
```

```
dimensiones correctas')
          else
            xBS(1:(nBS-1))=temp_vx;
            yBS(1:(nBS-1))=temp_vy;
            malladox=0:paso:15;
            malladoy=0:paso:15;
                for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                    aux_temp=(malladox-xBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
          end

       end



   else
       %si se han introducido xBs e yBS se comprueba que las longitudes
       %sean correctas
       if(length(xBS2)~=(nBS-1) || length(yBS2)~=(nBS-1))
          warning('los vectores yBS y xBS no tienen dimensiones correctas, como
quiere continuar?');
          eleccion=input('Introducir los vectores mediante dos vectores(v),
generarlos de forma aleatoria(a) o introducirlos graficamente(g)?: ','s');
          xBS=zeros(1,nBS);
          yBS=zeros(1,nBS);
          if(eleccion=='g') % se decide introducir los vectores de forma grafica
            [aux_xBS,aux_yBS]=ginput(nBS-1);
            xBS(1:(nBS-1))=aux_xBS;
            yBS(1:(nBS-1))=aux_yBS;
            malladox=0:paso:15;
            malladoy=0:paso:15;
            for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                aux_temp=(malladox-xBS(varaux_bucle)).^2;
                indice_auxliar=(aux_temp==min(aux_temp));
                xBS(varaux_bucle)=malladox(indice_auxliar);
```

```
                aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                indice_auxliar=(aux_temp==min(aux_temp));
                yBS(varaux_bucle)=malladoy(indice_auxliar);
            end

        elseif(eleccion=='a') %se decide que sea de forma aleatoria
            xBS(1:end-1)=randi([min(min(x))*2,max(max(x))*2],1,nBS-1)/2;
%fijamos las posiciones de nBS-1 estaciones base
            yBS(1:end-1)=randi([min(min(y))*2,max(max(y))*2],1,nBS-1)/2;

        elseif(eleccion=='v') %se decide que sea mediante un vector
            fprintf('introtuce un vector x de %d elementos',nBS-1);
            temp_vx=input('');
            fprintf('introtuce un vector y de %d elementos \n',nBS-1);
            temp_vy=input('');
            if(length(temp_vx)~=(nBS-1) || length(temp_vy)~=(nBS-1))
              error('alguno de los vectores introducidos no tiene las
dimensiones correctas')
            else
              xBS(1:(nBS-1))=temp_vx;
              yBS(1:(nBS-1))=temp_vy;
              malladox=0:paso:15;
              malladoy=0:paso:15;
                for varaux_bucle=1:(nBS-1) %decide aproximar las coordenadas
introducidas al punto mas próximo
                    aux_temp=(malladox-xBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    xBS(varaux_bucle)=malladox(indice_auxliar);
                    aux_temp=(malladoy-yBS(varaux_bucle)).^2;
                    indice_auxliar=(aux_temp==min(aux_temp));
                    yBS(varaux_bucle)=malladoy(indice_auxliar);
                end
            end

        end
    else
        xBS=[xBS2 0];
        yBS=[yBS2 0];

    end

end
```

```
    %se imprime sobre el escenario la solución inicial
    for ej2=1:nBS-1
      impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
      impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
      impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
      plot(impXBS,impYBS1);
      plot(impXBS,impYBS2);
    end
    plot(xBS(1:end-1),yBS(1:end-1),'*')
    for ej2=1:length(xBSfijos)
     impXBS=xBSfijos(ej2)-radiofijos(ej2):0.001:xBSfijos(ej2)+radiofijos(ej2);
     impYBS1=sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     impYBS2=-sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     plot(impXBS,impYBS1,'r');
     plot(impXBS,impYBS2,'r');
    end
    plot(xBSfijos,yBSfijos,'r*');

    hold off


%% empezamos calculando los cortes de las BS colocadas al principio
for ii=1:nBS-1 % todos los BS -1 fijos
    indice=ii*npaths-npaths+1; % cada Bs tiene tantas filas reservadas para ella
como paths hay
    for jj=1:npaths
      %comprobamos que la estación esta dentro del rango del segmento
      %para ahorrar tiempo
        if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-radio(ii)))
%comprobamos q esta dentro del rango del segmento

            if(xBS(ii)<(max(x(jj,:))+radio(ii)) && xBS(ii)>(min(x(jj,:))-
radio(ii)))
                        %rango de valores del eje x
                        valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
                        %segmento del path
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
                        %restamos los valores de y y buscamos cuando cambia
```

```
                    %de signo
                    aux=sign(valoresYBS1-valoresy);
                    aux(aux(1,:)~=1)=0;
                    aux2=sum(aux);
                    aux3=0;
                    if(aux2<length(aux) && aux2>0)
                        pos=find(aux(1)~=aux);
                        cortesx(indice,2)=jj;
                        cortesy(indice,2)=jj;
                        if(size(pos,2)~=0 && size(pos,2)~=1)
                            if(pos(end)==length(aux))
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesy(indice,3)=valoresYBS1(pos(1));
                                aux3=1;
                            else
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesx(indice,4)=valoresXBS(pos(end));
                                cortesy(indice,3)=valoresYBS1(pos(1));
                                cortesy(indice,4)=valoresYBS1(pos(end));
                                aux3=2;
                            end
                         end

                    end
                            % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
                    if(aux3<2)
                        aux=sign(valoresYBS2-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        if(aux3==0)
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                        end
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux) && aux2>0)
                                    if(aux3==1)
                                    %consideraciones aqui:
                                    %un path siempre va a cortar al radio en
dos puntos xq
```

```
                                    %esta hecho asi, por lo tanto comprobar
si un punto de
                                    %corte excede los limites del path y
ponerlo como el
                                    %punto extremo
                                    cortesx(indice,4)=valoresXBS(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(1));
                                    aux3=2;
                                    elseif (aux3==0)
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    aux3=1;
                                    end
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(end));
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
                        aux4=abs(xBS(ii)-x(jj,1));
                        aux5=abs(xBS(ii)-x(jj,2));
                        if(aux4==min([aux4 aux5]))
                            cortesx(indice,4)=x(jj,1);
                            cortesy(indice,4)=y(jj,1);
                        else
                            cortesx(indice,4)=x(jj,2);
                            cortesy(indice,4)=y(jj,2);
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)
```

```
socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=min(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                            else
                                cortesx(indice,4)=min(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                            end
                        end

                        if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                            if(socorro(1)==1)
                                cortesx(indice,3)=max(x(jj,:));
                                cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                            else
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                end % final de comprobacion del path

            end
          indice=indice+1;
        end
end

%hasta aqui los normales ahora los fijos

%llenamos las matrices con los id de las BS
for h=1:length(xBSfijos)
cortesx(nBS*npaths+1+(h-1)*npaths:nBS*npaths+npaths*h,1)=-1*h;
cortesy(nBS*npaths+1+(h-1)*npaths:nBS*npaths+npaths*h,1)=-1*h;
end

for ii=1:length(xBSfijos) % todos los BS -1 fijos
    indice=nBS*npaths+ii*npaths-npaths+1; % cada Bs tiene tantas filas
reservadas para ella como paths hay
    for jj=1:npaths % todos los paths
        %comprobamos que la estación esta dentro del rango del segmento
        %para ahorrar tiempo
```

```
        if(yBSfijos(ii)<(max(y(jj,:))+radiofijos(ii)) &&
yBSfijos(ii)>(min(y(jj,:))-radiofijos(ii))) %comprobamos q esta dentro del rango
del segmento


            if(xBSfijos(ii)<(max(x(jj,:))+radiofijos(ii)) &&
xBSfijos(ii)>(min(x(jj,:))-radiofijos(ii)))
                        %rango de valores del eje x
                        valoresXBS=xBSfijos(ii)-
radiofijos(ii):0.001:xBSfijos(ii)+radiofijos(ii);
                        %segmento del path
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        %valores y de la circunferencia radio de la BS
                        valoresYBS1=sqrt(radiofijos(ii)^2-(valoresXBS-
xBSfijos(ii)).^2)+yBSfijos(ii);
                        valoresYBS2=-sqrt(radiofijos(ii)^2-(valoresXBS-
xBSfijos(ii)).^2)+yBSfijos(ii);
                        %restamos los valores de y y buscamos cuando cambia
                        %de signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    cortesy(indice,4)=valoresYBS1(pos(end));
                                    aux3=2;
                                end
                             end


                        end
                                % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
```

```
                    if(aux3<2)
                        aux=sign(valoresYBS2-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        if(aux3==0)
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                        end
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux) && aux2>0)
                                    if(aux3==1)
                                    %consideraciones aqui:
                                    %un path siempre va a cortar al radio en
dos puntos xq
                                    %esta hecho asi, por lo tanto comprobar
si un punto de
                                    %corte excede los limites del path y
ponerlo como el
                                    %punto extremo
                                    cortesx(indice,4)=valoresXBS(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(1));
                                    aux3=2;
                                    elseif (aux3==0)
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    aux3=1;
                                    end
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesx(indice,4)=valoresXBS(pos(end));
                                    cortesy(indice,3)=valoresYBS2(pos(1));
                                    cortesy(indice,4)=valoresYBS2(pos(end));
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
```

```
                        aux4=abs(xBSfijos(ii)-x(jj,1));
                        aux5=abs(xBSfijos(ii)-x(jj,2));
                        if(aux4==min([aux4 aux5]))
                            cortesx(indice,4)=x(jj,1);
                            cortesy(indice,4)=y(jj,1);
                        else
                            cortesx(indice,4)=x(jj,2);
                            cortesy(indice,4)=y(jj,2);
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=min(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                        else
                            cortesx(indice,4)=min(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                        end
                    end

                    if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=max(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                        else
                            cortesx(indice,4)=max(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                        end
                    end
            end % final de comprobacion del path
```

```
        end
      indice=indice+1;
    end
end

% imprimimos la primera solucion
resultado1=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);

%% algoritmo principal
pause;

for ii=nBS:-1:1
    xBSdef=xBS(ii);
    yBSdef=yBS(ii);
    %la primera solución es dejarla en el mismo punto
    %esto se hace porque aunque cuando en el algoritmo vuelve a su posicion
    %no mejora nunca la solucion actual, entonces si ese es su optimo tiene
    %que quedarse ahi
   for gy=min(min(y)):paso:max(max(y))
       %recorremos el mallado
       for gx=min(min(x)):paso:max(max(x))

            %solo entra si no hay otra BS en ese punto
            if(sum(yBS(xBS==gx)==gy)==0) %solo entra si no hay otra BS en ese
punto
             yBS(ii)=gy;
             xBS(ii)=gx;
             %accedemos a la parte correspondiente de la matriz de cortes
             %ponemos la matriz de cortes correspondiente a la BS a 0
             indice=ii*npaths-npaths+1;
             cortesx(cortesx(:,1)==ii,3:4)=0;
             cortesy(cortesy(:,1)==ii,3:4)=0;

            for jj=1:npaths % todos los paths

                %comprobamos q esta dentro del rango del segmento
                if(yBS(ii)<(max(y(jj,:))+radio(ii)) && yBS(ii)>(min(y(jj,:))-
radio(ii))) %comprobamos q esta dentro del rango del segmento

                    if(xBS(ii)<(max(x(jj,:))+radio(ii)) &&
xBS(ii)>(min(x(jj,:))-radio(ii)))
                        %creamos el rango de valores del eje x
```

```
valoresXBS=xBS(ii)-radio(ii):0.001:xBS(ii)+radio(ii);
%calculamos los valores y de la recta
valoresy=pendiente(jj)*valoresXBS+origenY(jj);
valoresYBS1=sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
valoresYBS2=-sqrt(radio(ii)^2-(valoresXBS-
xBS(ii)).^2)+yBS(ii);
%restamos los valores de y y hallamos el punto
%donde se cambia el signo
aux=sign(valoresYBS1-valoresy);
aux(aux(1,:)~=1)=0;
aux2=sum(aux);
aux3=0;
if(aux2<length(aux) && aux2>0)
    pos=find(aux(1)~=aux);
    cortesx(indice,2)=jj;
    cortesy(indice,2)=jj;
    if(size(pos,2)~=0 && size(pos,2)~=1)
        if(pos(end)==length(aux))
            cortesx(indice,3)=valoresXBS(pos(1));
            cortesy(indice,3)=valoresYBS1(pos(1));
            aux3=1;
        else
            cortesx(indice,3)=valoresXBS(pos(1));
            cortesx(indice,4)=valoresXBS(pos(end));
            cortesy(indice,3)=valoresYBS1(pos(1));
            cortesy(indice,4)=valoresYBS1(pos(end));
            aux3=2;
        end
     end

end
        % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
if(aux3<2)
    aux=sign(valoresYBS2-valoresy);
    aux(aux(1,:)~=1)=0;
    aux2=sum(aux);
    if(aux3==0)
        cortesx(indice,2)=jj;
        cortesy(indice,2)=jj;
    end
```

```
                    if(aux2<length(aux) && aux2>0)
                        pos=find(aux(1)~=aux);
                        if(size(pos,2)~=0 && size(pos,2)~=1)
                            if(pos(end)==length(aux) && aux2>0)
                                if(aux3==1)
                                %consideraciones aqui:
                                %un path siempre va a cortar al radio en
dos puntos xq
                                %esta hecho asi, por lo tanto comprobar
si un punto de
                                %corte excede los limites del path y
ponerlo como el
                                %punto extremo
                                cortesx(indice,4)=valoresXBS(pos(1));
                                cortesy(indice,4)=valoresYBS2(pos(1));
                                aux3=2;
                                elseif (aux3==0)
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesy(indice,3)=valoresYBS2(pos(1));
                                aux3=1;
                                end
                            else
                                cortesx(indice,3)=valoresXBS(pos(1));
                                cortesx(indice,4)=valoresXBS(pos(end));
                                cortesy(indice,3)=valoresYBS2(pos(1));
                                cortesy(indice,4)=valoresYBS2(pos(end));
                                aux3=2;
                            end
                        end

                    end
                end

                if(aux3==1)
                    aux4=abs(xBS(ii)-x(jj,1));
                    aux5=abs(xBS(ii)-x(jj,2));
                    if(aux4==min([aux4 aux5]))
                       cortesx(indice,4)=x(jj,1);
                       cortesy(indice,4)=y(jj,1);
                    else
                       cortesx(indice,4)=x(jj,2);
                       cortesy(indice,4)=y(jj,2);
```

```
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=min(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                        else
                            cortesx(indice,4)=min(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                        end
                    end

                    if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=max(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                        else
                            cortesx(indice,4)=max(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                        end
                    end
                end % final de comprobacion del path

            end
          indice=indice+1;
        end

        %calculamos el resultado

resultado2=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);
        %si el resultado en este punto es mayor que el guardado,
        %sobreescribe los datos
        if(resultado2>resultado1)
            resultado1=resultado2;
```

```
                xBSdef=gx;
                yBSdef=gy;
                cortesxdef=cortesx((ii*npaths-npaths+1):(ii*npaths),3:4);
                cortesydef=cortesy((ii*npaths-npaths+1):(ii*npaths),3:4);
            end

            end


        end
    end
    %cuando llegamos al punto final guardamos la posicion del óptimo
    xBS(ii)=xBSdef;
    yBS(ii)=yBSdef;
    cortesx((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesxdef;
    cortesy((ii*npaths-npaths+1):(ii*npaths),3:4)=cortesydef;
    cortesxdef=zeros(npaths,2);
    cortesydef=zeros(npaths,2);

end


%%

        % HA PARTIR DE AQUI ES LA ULTIMA VERSIÓN

xBSad=zeros(1,exceso);
yBSad=zeros(1,exceso);
ii=0;
% se añaden estaciones adicionales hasta que se cumpla el porcentaje o se
% alcance el limite de estaciones
while((resultado1/sum(demand))<porcentaje && ii<exceso)

    ii=ii+1;

    for gy=min(min(y)):paso:max(max(y))
        % recorremos el mallado
        for gx=min(min(x)):paso:max(max(x))

            %solo entra si no hay otra BS en ese punto
            if(sum(yBSad(xBSad==gx)==gy)==0)
```

```
            yBSad(ii)=gy;
            xBSad(ii)=gx;
            indice=length(cortesx(:,1))-exceso*npaths+ii*npaths-npaths+1;
            %accedemos a la parte correspondiente de la matriz de cortes
            %ponemos la matriz de cortes correspondiente a la BS a 0
            cortesx(indice:indice+npaths,3:4)=0;
            cortesy(indice:indice+npaths,3:4)=0;


            for jj=1:npaths % todos los paths

                %comprobamos q esta dentro del rango del segmento
                if(yBSad(ii)<(max(y(jj,:))+max(radio)) &&
yBSad(ii)>(min(y(jj,:))-max(radio)))

                    if(xBSad(ii)<(max(x(jj,:))+max(radio)) &&
xBSad(ii)>(min(x(jj,:))-max(radio)))
                        %creamos el rango de valores del eje x
                        valoresXBS=xBSad(ii)-
max(radio):0.001:xBSad(ii)+max(radio);
                        %calculamos los valores y de la recta
                        valoresy=pendiente(jj)*valoresXBS+origenY(jj);
                        valoresYBS1=sqrt(max(radio)^2-(valoresXBS-
xBSad(ii)).^2)+yBSad(ii);
                        valoresYBS2=-sqrt(max(radio)^2-(valoresXBS-
xBSad(ii)).^2)+yBSad(ii);
                        %restamos los valores de y y hallamos el punto
                        %donde se cambia el signo
                        aux=sign(valoresYBS1-valoresy);
                        aux(aux(1,:)~=1)=0;
                        aux2=sum(aux);
                        aux3=0;
                        if(aux2<length(aux) && aux2>0)
                            pos=find(aux(1)~=aux);
                            cortesx(indice,2)=jj;
                            cortesy(indice,2)=jj;
                            if(size(pos,2)~=0 && size(pos,2)~=1)
                                if(pos(end)==length(aux))
                                    cortesx(indice,3)=valoresXBS(pos(1));
                                    cortesy(indice,3)=valoresYBS1(pos(1));
                                    aux3=1;
                                else
                                    cortesx(indice,3)=valoresXBS(pos(1));
```

```
                    cortesx(indice,4)=valoresXBS(pos(end));
                    cortesy(indice,3)=valoresYBS1(pos(1));
                    cortesy(indice,4)=valoresYBS1(pos(end));
                    aux3=2;
                end
            end

        end
        % HAY QUE CALCULAR LA PARTE DE ABAJO DE LA
CIRCUNFERENCIA
        if(aux3<2)
            aux=sign(valoresYBS2-valoresy);
            aux(aux(1,:)~=1)=0;
            aux2=sum(aux);
            if(aux3==0)
                cortesx(indice,2)=jj;
                cortesy(indice,2)=jj;
            end
            if(aux2<length(aux) && aux2>0)
                pos=find(aux(1)~=aux);
                if(size(pos,2)~=0 && size(pos,2)~=1)
                    if(pos(end)==length(aux) && aux2>0)
                        if(aux3==1)
                        %consideraciones aqui:
                        %un path siempre va a cortar al radio en
dos puntos xq
                        %esta hecho asi, por lo tanto comprobar
si un punto de
                        %corte excede los limites del path y
ponerlo como el
                        %punto extremo
                        cortesx(indice,4)=valoresXBS(pos(1));
                        cortesy(indice,4)=valoresYBS2(pos(1));
                        aux3=2;
                        elseif (aux3==0)
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesy(indice,3)=valoresYBS2(pos(1));
                        aux3=1;
                        end
                    else
                        cortesx(indice,3)=valoresXBS(pos(1));
                        cortesx(indice,4)=valoresXBS(pos(end));
                        cortesy(indice,3)=valoresYBS2(pos(1));
```

```
                                    cortesy(indice,4)=valoresYBS2(pos(end));
                                    aux3=2;
                                end
                            end

                        end
                    end

                    if(aux3==1)
                        aux4=abs(xBSad(ii)-x(jj,1));
                        aux5=abs(xBSad(ii)-x(jj,2));
                        if(aux4==min([aux4 aux5]))
                            cortesx(indice,4)=x(jj,1);
                            cortesy(indice,4)=y(jj,1);
                        else
                            cortesx(indice,4)=x(jj,2);
                            cortesy(indice,4)=y(jj,2);
                        end
                    end

                    %resolucion de consideraciones en caso de que solo corte
en
                    %un punto
                    if(min(cortesx(indice,3:4))<min(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==min(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=min(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==min(x(jj,:)));
                        else
                            cortesx(indice,4)=min(x(jj,:));
                            cortesy(indice,4)=y(jj,x(jj,:)==min(x(jj,:)));
                        end
                    end

                    if(max(cortesx(indice,3:4))>max(x(jj,:)) && aux3>0)

socorro=cortesx(indice,3:4)==max(cortesx(indice,3:4));
                        if(socorro(1)==1)
                            cortesx(indice,3)=max(x(jj,:));
                            cortesy(indice,3)=y(jj,x(jj,:)==max(x(jj,:)));
                        else
```

```
                                cortesx(indice,4)=max(x(jj,:));
                                cortesy(indice,4)=y(jj,x(jj,:)==max(x(jj,:)));
                            end
                        end
                    end % final de comprobacion del path

                end
              indice=indice+1;
            end

            end
            %calculamos el resultado

resultado2=optimiza_mio(x,y,cortesx,cortesy,npaths,demand,single,double,triple);
            %si el resultado en este punto es mayor que el guardado,
            %sobreescribe los datos
            if(resultado2>resultado1)
                resultado1=resultado2;
                xBSdef=gx;
                yBSdef=gy;
                cortesxdef=cortesx((length(cortesx(:,1))-
exceso*npaths+ii*npaths-npaths+1):(length(cortesx(:,1))-
exceso*npaths+ii*npaths),3:4);
                cortesydef=cortesy((length(cortesx(:,1))-
exceso*npaths+ii*npaths-npaths+1):(length(cortesx(:,1))-
exceso*npaths+ii*npaths),3:4);

            end

        end
    end
    %cuando llegamos al punto final guardamos la posicion del óptimo
    xBSad(ii)=xBSdef;
    yBSad(ii)=yBSdef;
    cortesx((length(cortesx(:,1))-exceso*npaths+ii*npaths-
npaths+1):(length(cortesx(:,1))-exceso*npaths+ii*npaths),3:4)=cortesxdef;
    cortesy((length(cortesx(:,1))-exceso*npaths+ii*npaths-
npaths+1):(length(cortesx(:,1))-exceso*npaths+ii*npaths),3:4)=cortesydef;
end


%% imprimir el mallado
  figure(2)
   hold on
   axis([0,15,0,15]);
```

```
    shg
for ej=1:npaths
    plot([x(ej,1) x(ej,2)],[y(ej,1) y(ej,2)],'k','LineWidth',3);
end
    shg

    %% imprimimos la posicion de las BS
    for ej2=1:nBS
     impXBS=xBS(ej2)-radio(ej2):0.001:xBS(ej2)+radio(ej2);
     impYBS1=sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     impYBS2=-sqrt(radio(ej2)^2-(impXBS-xBS(ej2)).^2)+yBS(ej2);
     plot(impXBS,impYBS1);
     plot(impXBS,impYBS2);
    end
    plot(xBS,yBS,'*');
    for ej2=1:length(xBSfijos)
     impXBS=xBSfijos(ej2)-radiofijos(ej2):0.001:xBSfijos(ej2)+radiofijos(ej2);
     impYBS1=sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     impYBS2=-sqrt(radiofijos(ej2)^2-(impXBS-xBSfijos(ej2)).^2)+yBSfijos(ej2);
     plot(impXBS,impYBS1,'r');
     plot(impXBS,impYBS2,'r');
    end
    plot(xBSfijos,yBSfijos,'*');

    % ii indica el numero de bases añadidas
    if(ii>0)
    for ej2=1:ii
     impXBS=xBSad(ej2)-max(radio):0.001:xBSad(ej2)+max(radio);
     impYBS1=sqrt(max(radio)^2-(impXBS-xBSad(ej2)).^2)+yBSad(ej2);
     impYBS2=-sqrt(max(radio)^2-(impXBS-xBSad(ej2)).^2)+yBSad(ej2);
     plot(impXBS,impYBS1,'g');
     plot(impXBS,impYBS2,'g');
    end
    plot(xBSad,yBSad,'g*');
    end
    hold off
    xBS=[xBS xBSad];
    yBS=[yBS yBSad];
    porcent=resultado1/sum(demand);
end
```