UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

Estimación Distribuida en Redes de Sensores Inalámbricos sin Centro de Fusión usando Modelos Paramétricos

AUTOR: María del Valle Nieto Blanco

TUTOR: David Luengo García

27 de julio de 2011

TÍTULO: *Estimación Distribuida en Redes de Sensores Inalámbricos sin Centro de Fusión usando Modelos Paramétricos.*

AUTOR: *María del Valle Nieto Blanco*

TUTOR: *David Luengo García*

La defensa del presente Proyecto Fin de Carrera se realizó el día 27 de julio de 2011; siendo calificada por el siguiente tribunal:

PRESIDENTE: *Luis Inclán Sánchez*

SECRETARIO *José Miguel Leiva Murillo*

VOCAL *David Griol Barres*

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

Agradecimientos

Este proyecto cierra un ciclo importante en mi vida y no quiero concluirlo sin antes dar gracias a todas aquellas personas que me han ayudado a llegar hasta aquí.

A mi tutor David, a quien le estoy muy agradecida por haberme brindado la oportunidad de realizar este proyecto, por guiarme, por su tiempo y, en especial, por compartir sus conocimientos conmigo.

A mis amigos y compañeros de la Universidad, que han compartido días de estudios y prácticas eternas, tanto en la biblioteca como en las salas de ordenadores. En especial a Dani por su paciencia y aguante.

Por último, agradecer a quienes les debo todo, mi familia. A mi tía Maribel porque sus cuidados tranquilizaron mi estancia en Madrid. A mis hermanos por estar ahí cuando los he necesitado, independientemente de la distancia. Con especial cariño y admiración, quiero destacar el apoyo de mis padres, por todos los años de esfuerzos y sacrificio para que pudiera optar a la mejor educación posible, por su paciencia, comprensión y los valiosos consejos que han sabido proporcionarme en lo profesional, pero, especialmente, en lo personal.

Muchas gracias a todos.

Sabe esperar, aguarda que la marea fluya,
así en la costa un barco, sin que al partir te inquiete
Todo el que aguarda sabe que la victoria es suya....
Antonio Machado

Resumen

En este Proyecto Final de Carrera se va a considerar la estimación distribuida en redes de sensores inalámbricas (WSNs) sin centro de fusión. El escenario que se va a plantear tiene como base una red de sensores distribuidos aleatoriamente dentro de un área limitada. Estos sensores son dispositivos inteligentes de bajo coste, bajo consumo de potencia y multifunción, capaces de detectar señales, procesar los datos recogidos de las mismas y establecer comunicación inalámbrica con otros terminales de la red directamente o a través de una estación base. Las WSNs tienen numerosas aplicaciones, entre las que se encuentran las medioambientales, a las que prestaremos especial atención en este proyecto. El desarrollo de las mismas se debe principalmente a los avances obtenidos en relación a la tecnología inalámbrica y a la miniaturización de los dispositivos electrónicos, permitiendo investigar fenómenos medioambientales en lugares, hasta ahora, inaccesibles.

En particular, en este proyecto analizaremos e implementaremos dos algoritmos de estimación distribuida (subgradiente y proyecciones alternas) que, a partir de los datos de medida proporcionados por una red de sensores, son capaces de estimar un conjunto de parámetros, correspondientes a una cierta aproximación funcional predeterminada, con los que podremos obtener una estima de la magnitud física real asociada a las medidas de dichos sensores. Para implementar estos algoritmos necesitaremos la ayuda de algún tipo de método de optimización centralizado (de la familia del gradiente) para la estimación de los parámetros. Además, ambos algoritmos necesitan disponer de una ruta a través de la red, confeccionada de tal modo que la información pase por todos los nodos una

sola vez, partiendo y regresando al mismo punto. Para la resolución de este problema se aplica alguno de los métodos disponibles para resolver el Problema del Viajante (TSP, "Traveling Salesman Problem"), habiéndose llegado a desarrollar versiones distribuidas de los mismos para su implementación eficiente en WSNs.

Finalmente, definiremos un modo de generar datos sintéticos que sustituirán a las medidas que proporcionaría una red de sensores real y expondremos los resultados obtenidos en las diferentes simulaciones llevadas a cabo con cada uno de los algoritmos de estimación distribuida, así como con los métodos distribuidos de resolución del TSP.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Motivación

En los últimos años el desarrollo de las Redes de Sensores Inalámbricas (WSNs, "Wireless Sensor Networks") ha despertado gran interés gracias a las numerosas aplicaciones y características interesantes que tienen. Las WSNs son capaces de recoger gran cantidad de información y almacenarla, de manera que se pueda utilizar para la mejora de sistemas militares, médicos, industriales, de predicción de desastres naturales, medioambientales, etc. Este último campo será en el que nos centremos en este Proyecto Final de Carrera, ya que, debido a los avances tecnológicos desarrollados en el siglo XX, que han permitido la miniaturización de los sensores, hoy en día podemos tomar medidas de magnitudes físicas en sitios que eran inaccesibles hasta hace unos pocos años.

Como acabamos de comentar, estas redes permiten el acceso a gran cantidad de datos, lo que nos lleva a la necesidad de procesar esa información, y así conocer en mayor profundidad la situación del entorno que estemos estudiando, proporcionando nuevas funcionalidades a los sistemas existentes hasta el momento, como pueden ser:

- Estimación de parámetros en una localización determinada. Esto nos permite obtener la medida de magnitudes físicas (temperatura, humedad, presión atmosférica, etc.) en un entorno de manera más local.
- Detección de la ocurrencia de eventos de interés y estimación de los parámetros de

los eventos detectados. Esta característica nos alertaría, por ejemplo, del cambio brusco de las magnitudes físicas medidas. En determinadas situaciones esto resulta fundamental, ya que una subida rápida de temperatura en un bosque podría significar la aparición de fuego en esa zona.

Teniendo en cuenta las necesidades que nos plantean los avances tecnológicos surgidos hasta el momento, resulta interesante buscar métodos que faciliten el procesamiento de tal cantidad de información. En este Proyecto Final de Carrera vamos a estudiar dos algoritmos de procesado distribuido con los que podremos estimar un conjunto de parámetros que nos ayudarán a encontrar una aproximación de la magnitud física real asociada a las medidas de los sensores.

1.2. Objetivos

En esta sección comentaremos los propósitos que queremos cumplir con el desarrollo de este Proyecto Final de Carrera. El *objetivo principal* es analizar, implementar y comparar mediante simulaciones dos algoritmos de procesado distribuido: el método del subgradiente y el de proyecciones alternas. Como *objetivos secundarios* tendremos los siguientes:

- Análisis de las WSNs en general y las medioambientales en particular, de forma que podamos conocer en mayor profundidad las características y funcionalidades de las mismas.

- Estudio comparativo del procesamiento de datos centralizado y distribuido, de modo que nos ayude a decidir cuál de ellos es el más adecuado para el problema que se plantea.

- Búsqueda de la ruta óptima con versiones distribuidas del problema del viajante (TSP, "Traveling Salesman Problem"), cuya finalidad es hacer que la información de la red pase por todos los nodos una sola vez, partiendo y regresando al mismo punto.

- Desarrollo de un mecanismo eficiente para la generación de datos sintéticos que, en ausencia de datos reales, permita simular las medidas de las magnitudes físicas tomadas por cada uno de los sensores que forman la red.

1.3. Estructura

El contenido de esta memoria está dividido en siete capítulos. El *primer capítulo*, que es en el que nos encontramos, está dividido, a su vez, en tres secciones, donde definiremos los motivos que nos han llevado a realizar este Proyecto Final de Carrera, los objetivos que deseamos cumplir y esta sección, en la que detallamos la organización del documento.

En el *segundo capítulo* analizaremos las Redes de Sensores Inalámbricas, describiendo la estructura general de la red, junto con los principales problemas y desafíos que nos podemos encontrar, así como las distintas aplicaciones de dichas redes. En la última sección nos centraremos en las Redes de Sensores Medioambientales, que serán el objeto de nuestro estudio, detallando cómo se lleva a cabo la comunicación entre nodos, algunos de los tipos de redes existentes y, nuevamente, los problemas y desafíos específicos de las mismas.

En el *tercer capítulo* confrontaremos el procesamiento centralizado y distribuido con el fin de ver cuál de ellos es el más adecuado para las Redes de Sensores Inalámbricas Medioambientales. Este capítulo se divide en 3 secciones principales. En la primera detallaremos las características y funcionalidades de los principales elementos de las redes de sensores que trabajan bajo un procesamiento centralizado. En la segunda definiremos dos tipos de procesamiento distribuido, a la vez que examinaremos las ventajas y desventajas de ambos. Para finalizar, en la última sección compararemos tanto el procesado centralizado como el distribuido.

En el *cuarto capítulo* explicaremos los algoritmos implementados, junto con los métodos de optimización necesarios para llevar a cabo la estimación. Una vez más, el capítulo estará dividido en tres secciones. En la primera sección expondremos los métodos de optimización centralizados, que nos ayudarán en la estimación de los parámetros de manera distribuida. Todos ellos pertenecerán a la familia del gradiente: el descenso de gradiente,

el gradiente conjugado y el gradiente conjugado escalado. A continuación, en la segunda estudiaremos en detalle el primero de los algoritmos distribuidos: el subgradiente. Por último, en la tercera examinaremos el segundo algoritmo de optimización distribuida: el de proyecciones alternas.

En el *quinto capítulo* analizaremos el Problema del viajante, dada la equivalencia del mismo con la búsqueda del camino óptimo a través de la red. Resolver este problema es necesario para el desarrollo de los algoritmos que acabamos de mencionar, ya que uno de los obstáculos con el que nos hemos encontrado es cómo encontrar la ruta de menor distancia posible para recorrer todos los nodos de la red una sola vez. Como en los capítulos anteriores, este también estará dividido en tres secciones. En la primera enunciaremos formalmente el problema del viajante, mientras que en la segunda y en la tercera detallaremos los dos métodos distribuidos aplicados para resolver este problema: vecino próximo e inserción. Además, en esta última sección propondremos una modificación del método de inserción con el fin de obtener un algoritmo cuyo coste computacional sea menor.

En el *sexto capítulo*, por un lado explicaremos detalladamente cómo hemos generado los datos sintéticos necesarios para la realización de las simulaciones, y por otro mostraremos los resultados obtenidos en las simulaciones llevadas a cabo con cada algoritmo. Este capítulo estará dividido en cuatro secciones. En primer lugar, definiremos el método seguido para crear los datos sintéticos. En segundo lugar, expondremos los resultados conseguidos al calcular la ruta óptima con cada uno de los métodos mencionados. En tercer lugar, mostraremos los resultados de las simulaciones del primero de los algoritmos distribuidos, el subgradiente, y a su vez lo compararemos con un método centralizado, el gradiente. Por último, comentaremos los resultados de las simulaciones del segundo de los algoritmos distribuidos, el de proyecciones alternas

En el *séptimo capítulo*, enunciaremos las conclusiones obtenidas a lo largo de la realización de este Proyecto Final de Carrera y algunas posibles líneas futuras a seguir.

Capítulo 2

Redes de sensores inalámbricas

2.1. Introducción

Dentro del estudio que vamos a realizar en este Proyecto Final de Carrera, una parte fundamental son las redes de sensores inalámbricas (WSNs, "Wireless Sensor Networks"). Estas redes se pueden definir como un conjunto de dispositivos inteligentes de bajo coste, bajo consumo de potencia y multifunción, que tienen la capacidad de detectar propiedades físicas, biológicas y químicas de su ambiente, convertirlas en señales eléctricas para procesar los datos recogidos, y establecer una comunicación inalámbrica con otros terminales de la red para la distribución de los datos procesados. Típicamente constan de un gran número de nodos densamente distribuidos, capaces de detectar y rastrear señales [7], [8].

Las WSNs tienen su origen en aplicaciones militares, las primeras de las cuales se desarrollaron durante la Guerra Fría [9]. Las investigaciones que se llevaron a cabo durante esa época hicieron posible la evolución de dichas redes en aspectos tales como el tamaño de los sensores, el tipo de comunicación o la distribución de los mismos, apareciendo numerosas aplicaciones civiles en años posteriores [8], [10], [11], [12], [13]. Actualmente las WSNs permiten realizar medidas de todo tipo de parámetros físicos, gracias al progreso en los circuitos digitales y en el mundo de la radiofrecuencia, que han convertido en una realidad la incorporación de transmisores de alta frecuencia y bajo consumo de potencia en los nodos sensores. Este tipo de dispositivos reciben el nombre de motas, que se suelen

definir como terminales autónomos y compactos [14]. Estas características, junto con el hecho de que estos terminales lleven un procesador a bordo, permiten la comunicación entre los nodos vecinos, de forma que puedan enviar la información recogida entre ellos, fusionarla y llegar a resultados coherentes. Con anterioridad, debe aprovecharse la capacidad de procesamiento de los mismos para simplificar la información disponible, transmitiendo sólo los datos que puedan resultar relevantes a sus vecinos, ya que las WSNs tienen requisitos muy estrictos de energía y comunicación.

Su éxito se debe, principalmente, a las numerosas aplicaciones que tienen. Son capaces de recoger gran cantidad de información y almacenarla, de manera que se pueda utilizar para la mejora de sistemas militares, médicos, industriales, de predicción de desastres naturales, medioambientales, etc [8], [10], [11], [12], [13].

El contenido de este capítulo está dividido en tres secciones, con las que intentaremos conocer en mayor profundidad las WSNs:

- En la Sección 2.2 mostraremos su estructura general y describiremos uno de los elementos más importantes de las mismas, como son los nodos.
- En la Sección 2.3 expondremos los problemas y desafíos relacionados con el procesamiento de los datos, la comunicación y el manejo de los sensores que estas redes nos pueden presentar.
- En la Sección 2.4 nos centraremos en las redes de sensores medioambientales, estudiando cómo debe llevarse a cabo la comunicación entre los distintos nodos, el procesamiento de datos, los tipos de redes que podemos encontrarnos dentro de este ámbito y los problemas y desafíos a los que podríamos tener que enfrentarnos a la hora de implantarlas en cualquier medio.

2.2. Estructura general de la red

En la Figura 2.1 se muestra una posible estructura general de una red de sensores jerárquica. Como vemos, está formada por un servidor, al que se conectan varias estaciones

base, a las cuales se conectan a su vez los diferentes nodos que conforman la red. Según las flechas de la figura, deducimos que en el nivel más bajo, es decir, en los nodos, la movilidad es generalmente mucho mayor que en el servidor de la red. Por otro lado, tenemos que en el nivel superior, el servidor, la potencia, la capacidad de memoria y el tamaño es típicamente mayor que en niveles inferiores.

Nótese que, en la arquitectura de la Figura 2.1, se permite la conexión entre nodos que estén conectados a una misma estación base, pero no se puede llevar a cabo la conexión directa entre nodos conectados a distintas estaciones base, ni la conexión entre estaciones base. Todo esto se tendría que hacer a través del servidor. Es importante resaltar el hecho de que existen otro tipo de configuraciones de red. Por ejemplo, aquellas que permiten la conexión directa entre todos los nodos que forman la red, como podremos ver más adelante.

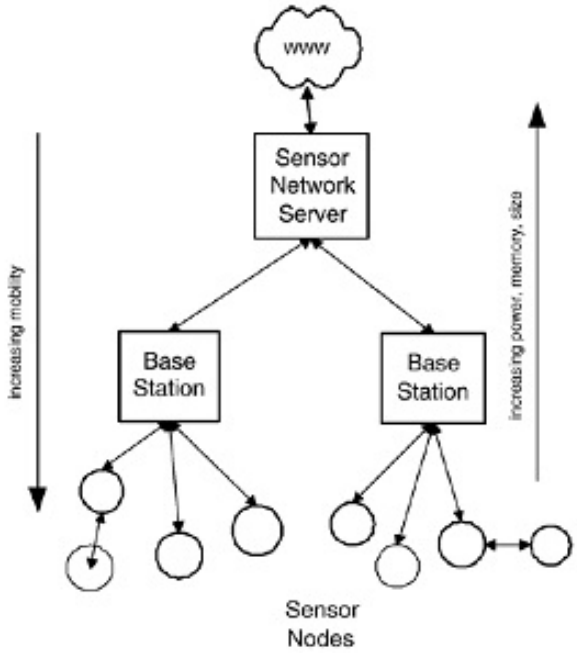


Figura 2.1: Posible estructura general de una red de sensores [1].

Una de las partes más importantes de la red son los nodos sensores, que, como se puede ver en la Figura 2.2, estarán formados por cuatro unidades principales [2]:

1. **Unidad de detección.** Compuesta por el sensor y un conversor analógico/digital

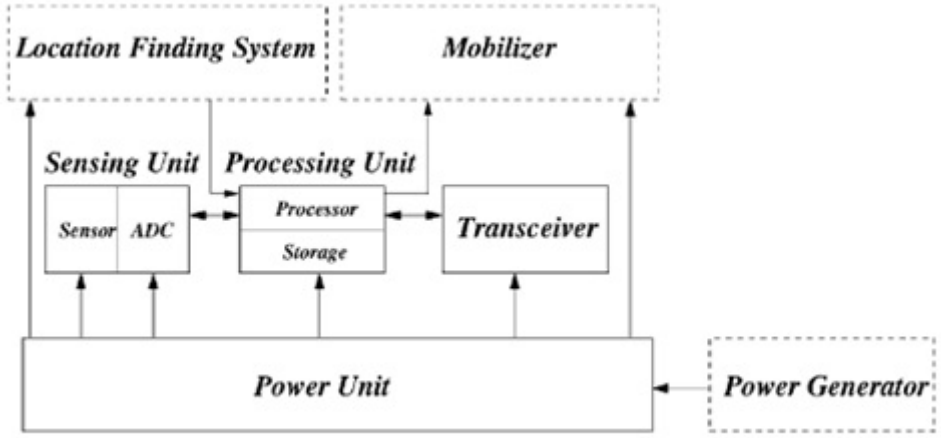


Figura 2.2: Componentes de un nodo sensor [2].

(ADC). El sensor genera una señal analógica que dependerá de lo que haya observado. Dicha señal se convierte a digital con ayuda del ADC.

2. **Unidad de procesamiento.** Formada por un procesador y una pequeña unidad de almacenamiento. El procesador recoge la señal digital del ADC y se encarga de que el nodo colabore con los nodos vecinos y lleve a cabo la tarea asignada. La unidad de almacenamiento guarda los datos temporalmente mientras sean necesarios para su procesado.

3. **Transceptor.** Conecta el nodo a la red y realiza las funciones de transmisión y recepción. Los nodos sensores hacen uso habitualmente de las bandas ISM (Industrial, Científica y Médica) [15], que proporcionan un gran ancho de banda y disponibilidad global. El medio de transmisión es típicamente las ondas de radiofrecuencia, aunque también se pueden utilizar las comunicaciones ópticas (láser) e infrarrojas. El láser necesita menos potencia, pero requiere visión directa y tiene más sensibilidad a las condiciones climatológicas. El infrarrojo, al igual que el láser, no necesita una antena. En la actualidad, la comunicación por radiofrecuencia es la que se utiliza en la mayoría de las aplicaciones de este tipo de redes [2].

4. **Unidad de alimentación.** Una de las más importantes del nodo. La potencia se

almacena en baterías o condensadores, que pueden ser recargables o no. La política de ahorro de energía se basa en técnicas tales como la administración dinámica de potencia (DPM, "Dynamic Power Management") y el escalado dinámico de tensión (DVS, "Dynamic Voltage Scaling") [16].

2.3. Problemas y desafíos

Las redes de sensores actuales presentan una serie de problemas relacionados con el procesamiento de datos, la comunicación y el manejo de sensores [9]. Por un lado, las WSNs habitualmente son redes "ad hoc", que no tienen una topología de red conocida a priori, sino que se va construyendo en tiempo real. Una de las ventajas de estas redes es que los nodos que las componen pueden comunicarse entre ellos y, a su vez, compartir los datos recogidos. Sin embargo, para llevar a cabo esta tarea deben conocer la posición e identidad de sus vecinos, además de la propia. Esto supone una desventaja importante, ya que la topología de las mismas es desconocida a priori. También debemos tener en cuenta que, tanto las redes como los dispositivos con los que debemos trabajar, necesitan una serie de recursos determinados, como son potencia de procesamiento, energía y ancho de banda. La necesidad de estos va cambiando dinámicamente, por lo que sería deseable que los sistemas trabajaran de manera autónoma, pudiendo cambiar su configuración según las necesidades del momento. En consecuencia, la conectividad en las redes "ad hoc" va a depender del software y los algoritmos que se empleen [17].

Por otro lado, en anteriores ocasiones se ha mencionado que todos los nodos de la red recogen información de interés de su entorno y la comparten con sus vecinos. La dificultad que se nos plantea aquí es cómo fusionamos todos los datos que recibe un nodo y cómo discriminamos la información relevante de la no relevante. A la hora de fusionar esta información podemos usar desde las reglas más simples hasta las técnicas más sofisticadas, que tengan en cuenta cómo se generaron los datos. Por ejemplo, las señales que llegan a los nodos pueden haber pasado por múltiples caminos, por lo que debería evitarse almacenar información repetida. Para resolverlo podríamos pensar en guardar alguna característica relevante de las señales recibidas, para, en el caso de volver a recibir la misma señal, poder

identificarla como repetida. Desafortunadamente, al trabajar con redes "ad hoc", donde los sensores que las componen tienen capacidad de almacenamiento limitada, esto no resulta factible. Además, estas redes se utilizan habitualmente en la detección de múltiples objetivos, con lo que cada nodo debería asociar los datos recogidos con su objetivo, pero la asociación de datos y objetivos óptima es costosa computacionalmente y requiere un gran ancho de banda [18].

Por último, una red de sensores puede verse como una base de datos con características únicas, que continuamente está adquiriendo información de su entorno. Estos datos son distribuidos hacia los nodos vecinos, conectados entre ellos por enlaces no fiables, característica no deseable para aplicaciones como las militares o médicas, donde la seguridad y la fiabilidad de los datos se consideran parámetros críticos para el buen funcionamiento de las redes. En estos y otros casos resulta de gran relevancia que el usuario disponga de una interfaz simple para interactuar de manera robusta con dicha red. Por consiguiente, uno de los desafíos actualmente es desarrollar un lenguaje de pregunta y respuesta que haga de intermediario entre el usuario y la base de datos. Otro de los retos con los que nos encontramos es la búsqueda de un mecanismo eficiente para la recopilación, organización y almacenamiento de datos [18].

2.4. Redes de sensores medioambientales

Son múltiples las aplicaciones que encontramos hoy en día para las redes de sensores: domóticas, de seguridad, industriales, militares, médicas, medioambientales, etc. En este Proyecto Final de Carrera nos centraremos en las aplicaciones medioambientales, que comentaremos con mayor detalle en este apartado. El desarrollo de las redes de sensores medioambientales se debe, principalmente, a los avances obtenidos en relación a la tecnología inalámbrica y a la miniaturización de los dispositivos electrónicos, que van a permitir el estudio de fenómenos medioambientales en lugares inaccesibles hasta ahora. Estas redes facilitan el estudio de procesos fundamentales en el medioambiente y permiten desarrollar sistemas capaces de responder ante ciertas alertas o peligros. En las últimas décadas, han ido evolucionando desde los registradores automáticos, donde la descarga de

los datos era manual, hasta los nodos sensores inteligentes, que comunican la información almacenada de forma autónoma a una red de servidores [1]. En cuanto a los nodos, pueden ser fijos o móviles, y sus dimensiones acordes al medio que les rodea. Además, debemos tener en cuenta que el sistema a instalar tiene que ser capaz de soportar las condiciones climatológicas del medio [1]. A continuación se detallarán algunos aspectos específicos de este tipo de redes.

2.4.1. Comunicación

Las comunicaciones son una parte esencial de las WSNs en general, así como de las redes medioambientales en particular. Por eso, hemos decido centrar esta sección en algunos aspectos importantes de las mismas, como son la transmisión y la gestión de potencia. Esta comunicación típicamente debe hacerse bajo las condiciones impuestas por el estándar IEEE 802.15.4, que fue definido para el intercambio de información de bajo coste y consumo en redes de área personal, que incluyen redes de sensores para la monitorización de datos y el control de actuadores [19].

Respecto a la transmisión, debemos tener en cuenta que podemos encontrarnos con dificultades a la hora de transmitir, puesto que en el medio con el que trabajaremos nos podremos topar con hielo, vegetación, etc. Además, en general sabemos que, cuanto mayor sea la frecuencia mayor necesidad tendremos de disponer de visión directa entre las estaciones y mayores serán las pérdidas en ambientes húmedos y con vegetación espesa. En este caso, las frecuencias utilizadas están entre 433 y 868 MHz en Europa, lo que implica que podemos alcanzar una distancia de transmisión como mucho de aproximadamente 200 m, debido a la regulación de potencia. Esta es la razón por la que las redes de sensores actuales tienen un área de cobertura entorno al kilómetro cuadrado como máximo. Afortunadamente, como hemos comentado anteriormente, es posible enlazar una serie de nodos a través de redes “ad hoc”, donde el salto entre nodos permite extender el rango de la red. Alternativamente, se pueden usar estaciones base/pasarelas intermedias.

Respecto a la gestión de la red, existen varios enfoques para la creación y control de este tipo de redes. Un enfoque orientado a la gestión de potencia, usado habitualmente,

es configurar los nodos para "despertarlos" durante cortos períodos de tiempo, en los que recogeremos las lecturas realizadas por los mismos. Por esto, deberemos distinguir entre nodos pasivos y activos [1]:

- **Nodos pasivos.** Son interrogados secuencialmente por la estación base. Con esto se reduce la complejidad, además de evitar la competición para acceder al medio.

- **Nodos activos.** Envían datos fuera de la red a través de las pasarelas. Requieren protocolos de red más sofisticados para evitar colisiones cuando existen varios nodos intentando transmitir.

Otro tipo de enfoque estaría orientado a la distribución de los nodos, ya que, según el tipo de información que queramos recoger, nos podrá interesar situar más o menos nodos en determinadas zonas. Por ejemplo, en el caso de querer monitorizar la actividad o evolución de ciertas aves, lo lógico sería desplegar la red de sensores en los nidos de las mismas o en las proximidades. Por otro lado, también deberemos tener especial cuidado en utilizar una tecnología que sea lo menos invasiva posible, para asegurar que la información recogida sea realista.

Con esto, podemos concluir que las características de la aplicación y los nodos van a determinar la técnica a usar. No obstante, una buena alternativa, utilizada frecuentemente, consiste en definir un protocolo basado en ranuras ("slots") temporales que permitirá a múltiples nodos comunicarse, evitando las colisiones con ayuda de buenos tiempos de sincronización. Por ejemplo, Jet Propulsión Lab's Sensor Webs usa este tipo de técnicas, obligando a los nodos a almacenar sus propios datos y los de sus vecinos, lo que proporciona dos ventajas [20]:

- Los nodos pueden cambiar su comportamiento en función de los datos almacenados localmente.

- Aseguramos el camino de los datos hacia su destino final.

2.4.2. Procesamiento de datos

La evolución de estas redes ha traído consigo mejoras en el manejo de los datos y el acceso a los mismos. La cantidad de información generada por las mismas no se había visto antes en ninguna rama de la geociencia, por lo que requieren un sistema propio de tratamiento de datos, almacenamiento y visualización. La mayoría de estos sistemas no permiten la recogida automática de información por parte de los usuarios, pero cuentan con unos motores de búsqueda que incluyen descarga de datos desde una Web. Los servicios Web nos suministran almacenamiento distribuido y capacidades de procesamiento, así como recuperación de datos. De esta idea surgió el concepto de Ciberinfraestructura, acuñado por el comité de la Fundación Nacional de las Ciencias (NSF, "National Science Foundation") de EE.UU. [21]. Con él querían describir el nuevo entorno de investigación, en el que los servicios de cálculo avanzado, adquisición y gestión de datos estaban disponibles para los usuarios a través de redes de alto rendimiento. Dos ejemplos de ciberinfraestructura son GRID y la Web semántica [22], que no sólo facilitan los recursos básicos para cálculos de gran escala, sino que, además, proporcionan la infraestructura necesaria para la gestión de los datos y el software.

Desde que surgieron estas redes, la cantidad de información que se ha ido almacenando en las bases de datos es de tamaño considerable, estando muchas de ellas disponibles a través de Internet. Sin embargo, si un investigador quisiera recopilar toda esta información, tendría que visitar cada sitio manualmente y, a su vez, transformar los archivos a un formato útil. El desafío en este sentido radica en la necesidad de crear unas normas para la publicación de datos y una interfaz que facilite la recopilación de los mismos. Una posible solución sería el lenguaje de marcado extensible (XML, "Extensible Markup Language"), metalenguaje extensible de etiquetas que permite definir la gramática de lenguajes específicos [23]. Aún así, es posible que cada proyecto defina un esquema completamente diferente y los problemas de integración de datos continúen. El lenguaje de marcado geográfico (GML, "Geography Markup Language") es un sublenguaje de XML, descrito como una gramática para el modelado de transporte y almacenamiento de información geográfica, que incluye información geoespacial y temporal [24]. Este tipo de

metadatos estándar es un elemento fundamental en la ciberinfraestructura. Sin embargo, también resulta de vital importancia definir un protocolo para el envío y recepción de mensajes, principalmente para emergencias. Para ello, se ha definido el Protocolo Común de Alertas (CAP, "Common Alerting Protocol") [25]. CAP y GML se usan en el proyecto conocido como SensorNet [26].

Otro proyecto interesante es "GEOsciences Network" (GEON) [27], que tiene como objetivo crear un prototipo para la interpretación del entorno mediante el uso de tecnologías que faciliten la colaboración entre las distintas ciencias. Está basado en el uso de motores de búsqueda, herramientas de consulta de la cartografía (GIS, "Geographic Information System") y herramientas de visualización 3D [28]. Algunas de sus funcionalidades son el acceso autentificado a datos y servicios Web, tres niveles de registro para los datos, herramientas y servicios, así como la facilitación de la búsqueda de los mismos y la creación de un entorno de procesamiento de datos científico.

2.4.3. Tipos de redes

Dentro de las redes de sensores medioambientales tenemos cuatro tipos diferentes, que describiremos brevemente en los puntos que siguen [1]:

1. **Redes de gran escala.** Toman medidas con un único objetivo. Suelen utilizar una red con topología en estrella, en vez de las redes "ad hoc", y sus nodos son de gran tamaño y bastante caros. Normalmente se conectan a la red eléctrica y tienen aplicaciones que van desde una simple estación meteorológica hasta la Red Mundial Sismográfica (GSN, "Global Seismographic Network) [29].

2. **Redes multifunción localizadas.** Toman medidas genéricas de propiedades meteorológicas y medioambientales. En este caso las redes suelen ser "ad-hoc". Las ventajas de estos sistemas son la simplicidad de implementación y los nodos, que se pueden conseguir con facilidad, ya que están disponibles comercialmente, y se conocen como motas. Estos nodos almacenan y comparten los datos, pudiendo cambiar su comportamiento en base a la información almacenada. Sistemas parecidos a estos

se usan en los edificios para controlar la humedad y la ventilación de los mismos [30].

3. **Redes de biosensores.** Se distinguen de las dos anteriores principalmente por el uso de la biotecnología. Un biosensor está compuesto por un sensor biológico conectado a un transductor físico, que puede ser acústico, electroquímico, óptico u optoelectrónico. Los agentes contaminantes que estudian este tipo de redes no son monitorizados en el lugar donde se recolectan las muestras, sino que son analizados en el laboratorio. Estos dispositivos almacenan información ante la presencia de diferentes agentes biológicos o químicos en el medioambiente. En general, las investigaciones en este campo están orientadas al desarrollo de sensores de menor tamaño y que sean capaces de detectar contaminantes en medios líquidos, gaseosos o sólidos, así como a monitorizar microorganismos en dichos medios de manera fiable [31].

4. **Redes heterogéneas.** Resultan cada vez más populares, ya que permiten incrementar el tiempo de vida y fiabilidad de las redes sin un gran aumento en el coste de las mismas. Se pueden diseñar dos tipos de redes: unas, cuyo objetivo es la detección, que suelen estar formadas por un gran número de nodos de bajo coste, y otras, formadas por pocos nodos de mayor coste, cuyo objetivo es filtrar, fusionar y transportar datos. Este tipo de redes incluyen las fuentes de datos de las tres clases de redes que hemos descrito hasta ahora, y sirven para monitorizar el medioambiente en diferentes escalas. Las medidas pueden ser automáticas o manuales, la mayoría de los nodos de medida tienen una única función, y dichos nodos pueden ser enlazados para formar una red inteligente. A modo de ejemplo, el Sistema Mundial de Observación de la Tierra (GEOSS, "Global Earth Observation System of Systems") planea combinar datos procedentes de las boyas oceánicas, estaciones medioambientales y satélites [32].

En la Figura 2.3 podemos observar un esquema de los tipos de redes que acabamos de definir, así como la relación que existe entre ellas.

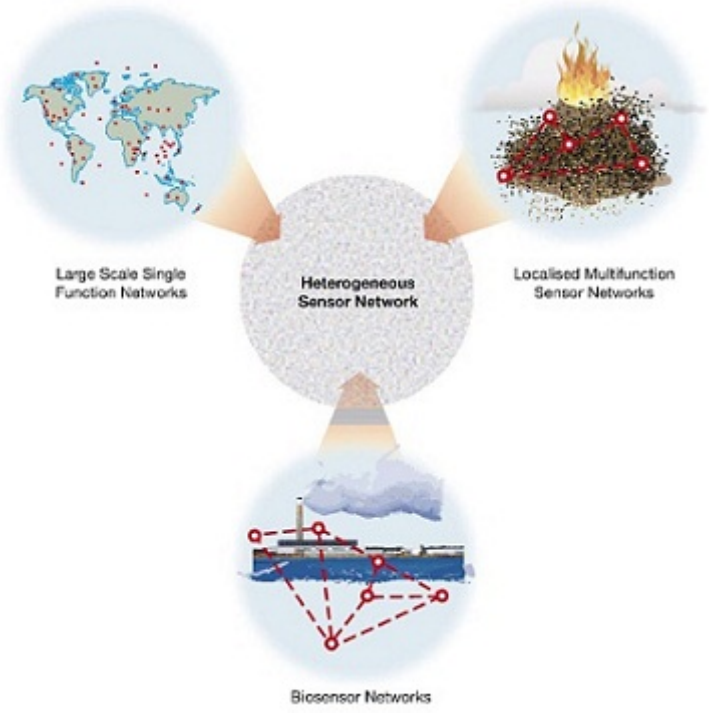


Figura 2.3: Diagrama esquemático de las distintas clases de redes de sensores medioambientales consideradas [1].

2.4.4. Problemas y desafíos

En este apartado vamos a centrarnos en los problemas y desafíos que suponen las redes de sensores aplicadas en un entorno medioambiental [1]:

- **Calidad de los datos.** En relación con este aspecto, podemos decir que la calibración de los nodos es muy importante, ya que una mala calibración puede deteriorar la calidad de la información. Además, sería deseable poder conocer el origen de la misma y, así, tener la posibilidad hacer un mejor análisis.

- **Comunicaciones radio.** En la mayoría de los casos, los agentes meteorológicos impredecibles pueden hacer que perdamos la comunicación entre los nodos o entre las estaciones. En un futuro, será necesario el uso de transmisores capaces de alterar la potencia de emisión para paliar este efecto.

- **Estandarización.** En la actualidad existen muchos fabricantes de equipos y cada

uno utiliza su propio software interfaz. Esto implica que la interoperabilidad entre los mismos es muy costosa y, en ocasiones, imposible. El desafío para un futuro es poder formar una red de nodos, donde cada uno de ellos pueda ser de un fabricante distinto y que la información llegue a un destino común de una manera coherente.

- **Gestión de potencia.** El objetivo es tener nodos que no necesiten el continuo recambio de baterías porque estas se agoten. Es necesario el desarrollo de técnicas sofisticadas para el manejo de la potencia. Extrapolando el problema al sistema GlacsWeb [33], que consiste en una red de sensores medioambientales instalados en 2004 en glaciares en Briksdalsbreen (Noruega) y cuya finalidad era estudiar la evolución de dichos glaciares, observamos que se intentan usar esquemas de planificación de potencia, que permitan reducir el consumo de potencia, aumentando la eficiencia. Por el hecho de querer garantizar cierta fiabilidad en las comunicaciones de gran alcance se han utilizado esquemas de gestión adaptativa de potencia, pero si se usaran redes "ad hoc" con comunicación entre nodos sensores se podría reducir el consumo [34].

- **Gestión remota.** Este tipo de redes se instalan en lugares aislados, a los que no se puede acceder con cierta regularidad. No obstante, se necesita acceder remotamente a ellas para corregir errores, solucionar caídas de los subsistemas, etc. Para llevar a cabo este tipo de gestión, el sistema Glacsweb, por ejemplo, instaló una webcam en la estación base, de modo que se pudiera vigilar el estado físico de la instalación en todo momento [33].

- **Miniaturización.** Está relacionada con el hecho de que estas redes deben ser no invasivas, de modo que cuanto menor sea el tamaño de los nodos sensores mucho mejor para conseguir este objetivo.

- **Seguridad.** Es importante para evitar la manipulación o pérdida de información, tanto a nivel hardware como a nivel de gestión. Se podrían usar protocolos de red convencionales, pero el problema es que consumen demasiada potencia. En consecuencia, será necesario desarrollar protocolos de seguridad de bajo consumo.

- **Usabilidad.** Sería deseable que las redes que se implantasen utilizaran técnicas fáciles de comprender y usar. Sin embargo, debido a que estos sistemas se han desarrollado bajo plataformas de investigación, se requieren altos conocimientos técnicos. El sistema GlacsWeb, por ejemplo, no podría ser instalado por muchos científicos porque, para ello, es necesario tener un gran conocimiento sobre ordenadores y electrónica, sin contar con que dispone de interfaces complejas de manejar [33]. En este sentido, sería de gran ayuda aplicar tecnicas de "plug-and-play" en este ámbito. Otra herramienta, que podría ayudar a los científicos es BeanWatcher [35], que direcciona de forma semiautomática la monitorización y la gestión.

2.5. Discusión

Después del estudio realizado sobre las redes de sensores, podemos concluir que, aunque tienen numerosas aplicaciones, aún deben mejorarse muchos aspectos tecnológicos, como podría ser la seguridad de las mismas, el procesamiento de datos y la comunicación entre los nodos de las redes. Por otro lado, podemos decir que suponen un gran avance, ya que una de las principales características de estas redes es que son capaces de recopilar mucha información. Estos datos pueden servirnos para realizar numerosos estudios que podrían ayudarnos a evitar, en un futuro, ataques terroristas, militares o nucleares, o facilitar el camino a los investigadores para encontrar curas a ciertas enfermedades y otras cuestiones.

En este estudio, además, hemos analizado las redes de sensores medioambientales. Actualmente, una de las mayores preocupaciones es el calentamiento global de La Tierra. El estudio de los agentes contaminantes en las ciudades, la evolución de las especies, ríos, mares y océanos pueden servirnos de gran utilidad para encontrar soluciones, de forma que reduzcamos, en la medida de lo posible, los efectos del cambio climático. En este sentido, en los últimos años se han desarrollado numerosas redes de sensores medioambientales (GlacsWeb, SensorNet, GEOSS, etc), que permiten analizar todo tipo de características.

Capítulo 3

Procesado distribuido vs. centralizado

3.1. Introducción

En el Capítulo 2 hemos comentado que las WSNs generan gran cantidad de información, que tendremos que procesar para obtener resultados coherentes. El problema que aquí intentaremos resolver es qué tipo de procesado es el más adecuado para estas redes: centralizado o distribuido.

Considérese el siguiente escenario [4]. Se dispone de un conjunto de sensores inalámbricos situados en un espacio $X \subset R^2$, con $\boldsymbol{x}_i = [x_{1,i}\ x_{2,i}]^T \in X$ las coordenadas del sensor i-ésimo, es decir, su posición. Cada uno de estos sensores medirá una magnitud física (por ejemplo, temperatura, presión o humedad), de modo que la observación del sensor i-ésimo es $y_i \in Y \subset R$. En resumen, los sensores se desplegarán en el medio y adquirirán colectivamente una serie de datos, que serán nuestras medidas y representarán la magnitud física que nosotros queramos medir. Bajo estas condiciones, nuestro objetivo será conocer las medidas en cualquier punto del espacio de observación.

El procesamiento de los datos detectados por un sensor nos da una idea de las propiedades de los objetos localizados alrededor del mismo, de la magnitud a medir y/o de los eventos que ocurren en la vecindad, según la aplicación. Una red de sensores puede contener cientos o miles de nodos, que nos permiten la detección de las magnitudes deseadas en grandes regiones geográficas, gracias a la coordinación entre sí de los nodos,

obteniendo así información de alta calidad del medio que les rodea. Una vez que los nodos han terminado de obtener las medidas, deben llevar a cabo el procesamiento de esa información, teniendo en cuenta sus limitaciones en cuanto a energía y ancho de banda. Para realizar un buen procesado de los datos, deberemos considerar, entre otros, los siguientes aspectos [2], [36]:

- El conocimiento de la posición es importante para la recogida de datos, ya que tanto la recolección de los mismos como su procesado puede depender de la situación de los sensores. En general, las redes de sensores no pueden usar Sistemas de Posicionamiento Global (GPS, "Global Positioning System") debido a su excesivo coste, de modo que se usan métodos de posicionamiento más sencillos, que permiten a los nodos obtener una aproximación de su posición relativa. Como se ve en la Figura 3.1, existen dos clases de métodos: centralizados y distribuidos. El primero partirá de un único nodo central, que se encargará de realizar todas las tareas de cálculo y estimación de las posiciones absolutas o relativas de los distintos nodos que forman la red, usando la información global para mejorar dichas estimas. El segundo no trabajará con un sólo nodo central, sino que cada uno de los nodos de la red tendrá capacidad de cálculo y comunicación, estimando entre todos las posiciones relativas de cada uno y balanceando la carga de la red entre ellos [37].

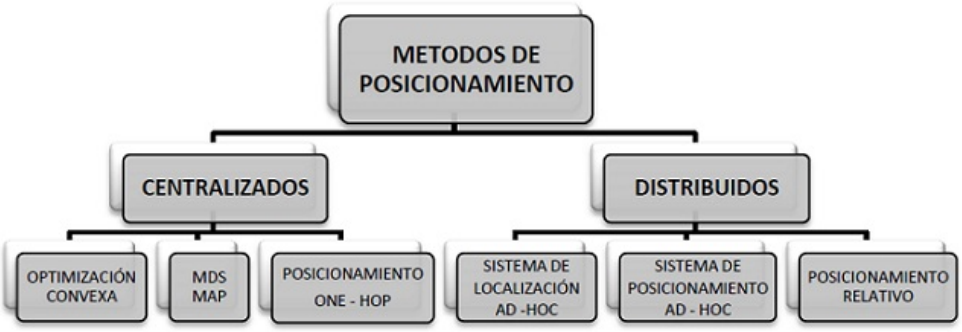


Figura 3.1: Métodos de posicionamiento [3].

- El hecho de que estas redes estén formadas por un gran número de sensores, ha impedido que se pueda construir un esquema de direccionamiento global, como el

existente en las pilas de protocolos basadas en IP, debiendo recurrirse con frecuencia a esquemas de direccionamiento basados en la posición.

- La mayoría de las aplicaciones necesitan que el flujo de datos discurra desde múltiples fuentes hacia una estación base (comunicación multi-punto).
- Las limitaciones de los nodos sensores hacen que deba buscarse la eficiencia desde el punto de vista del consumo de energía, procesamiento y almacenamiento, lo cual requiere un cuidadoso manejo de los recursos de los que disponemos.
- Los nodos habitualmente son estacionarios, aunque también nos podemos encontrar con redes formadas por nodos móviles o, incluso, híbridas.
- Los datos recogidos en diferentes sensores pueden deberse a fenómenos comunes, lo que implicaría la existencia de redundancia.

Con el fin de conocer en mayor profundidad tanto el procesamiento centralizado como el distribuido, hemos dividido este capítulo en tres secciones:

- En la Sección 3.2 trataremos de definir en qué consiste el procesado centralizado y cuáles son los problemas que nos podemos encontrar a la hora de implementarlo en las WSNs.
- En la Sección 3.3 definiremos el procesado distribuido y cuáles son los problemas que nos podemos encontrar a la hora de implementarlo en las WSNs. Dentro de este procesado estudiaremos dos topologías de redes: con centro de fusión y "ad-hoc".
- En la Sección 3.4 compararemos los dos tipos de procesado comentados anteriormente.

3.2. Procesamiento centralizado

Considérese un centro de fusión o estación base conectada a una red de N sensores, distribuidos aleatoriamente en un área geográfica determinada, en la que cada uno adquirirá

un valor de medida u observación. Suponemos que la comunicación entre el centro de fusión y los sensores es directa, es decir, que dichos sensores sólo establecerán comunicación con el centro de fusión mencionado, que será el encargado de recoger todos los datos sin cuantificación (esto es, con resolución infinita) y procesarlos, de manera que al final obtengamos una respuesta única. En la Figura 3.2 podemos ver un ejemplo del tipo de red que estamos considerando. El nodo rojo es el centro de fusión, mientras que los nodos azules son los sensores, que forman parte de la red, y las rayas punteadas simulan los enlaces.

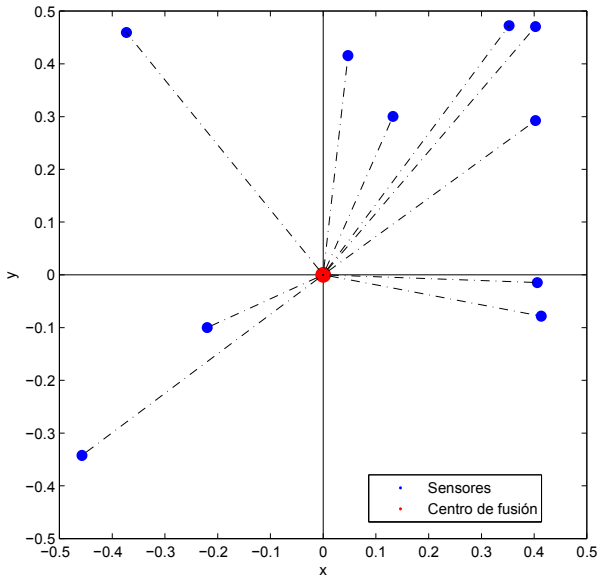


Figura 3.2: Ejemplo de una red de 10 sensores, distribuidos aleatoriamente dentro del área de observación, con procesamiento centralizado.

Dado que el centro de fusión será el encargado de llevar a cabo las tareas principales y de mayor coste, es lógico pensar que el hardware que necesitará será caro y de gran potencia. Los sensores, en cambio, sólo se dedicarán a recopilar la información del medio que les rodea y enviarla al centro de fusión. Por lo tanto, el hardware que estos necesitarán será más barato y tendrán las características definidas en el Capítulo 2.

Una de las tareas con que nos encontramos en el centro de fusión es la del manejo de memoria. Este deberá asignar la memoria de la que disponga a cada uno de los sensores a los que se encuentre conectado, para así poder almacenar la información que estos le

envíen. Una vez almacenada la misma, deberá gestionarla, y para ello tendrá que asegurarse que la información recibida no está corrompida, ni deteriorada, facilitando el acceso por parte de los usuarios finales a la misma.

Como hemos comentado, las funciones de los nodos serán medir la magnitud física indicada y enviarla al centro de fusión, sin antes realizar ningún tipo de procesado sobre la información obtenida. Esto supone un elevado coste energético y de ancho banda necesario para la transmisión, que se convertirá en el principal problema de este tipo de procesado. Sin embargo, no será el único al que deberemos enfrentarnos, ya que, si se alcanza la máxima capacidad del centro de fusión, deberemos cambiar el hardware para aumentar la capacidad de cálculo y memoria. Esto también conlleva que la capacidad de cómputo esté limitada por el centro de fusión. Además, la comunicación con el centro de fusión puede llegar a ser un cuello de botella en el momento en que todos los sensores quieran acceder simultáneamente al mismo.

Actualmente, este tipo de procesado no suele utilizarse en redes de sensores inalámbricas, especialmente cuando el número de sensores es elevado. Es más común encontrarlo en entornos científicos, donde el objetivo principal es la ejecución de múltiples aplicaciones que sean capaces de llevar a cabo operaciones de cálculo mucho más costosas de las que podría realizar un ordenador común [38].

3.3. Procesamiento distribuido

Considérese ahora una red de sensores en la que nigún nodo dispone de toda la información. En esta clase de redes resulta más conveniente llevar a cabo un procesado distribuido o descentralizado, pudiendo distinguirse dos casos en función de la topología de la red: con centro de fusión y “ad-hoc” [4]. Ambos tipos se describen a continuación.

3.3.1. Procesamiento distribuido en redes con un centro de fusión

Este tipo de procesamiento tiene cierto parecido con el procesamiento centralizado, que discutimos en la sección anterior. Los nodos se comunican directamente con el centro

de fusión o puerta de enlace de forma paralela, tal y como se muestra en la Figura 3.3. Esto implica que la información que recibe dicho centro es superior, comparada con el procesado completamente distribuido que veremos en el siguiente apartado, y, por lo tanto, la capacidad de almacenamiento necesaria será mucho mayor. Esto puede ser un cuello de botella, y además requiere mayor consumo de energía, de modo que el tiempo de vida de la red será menor que el conseguido con el esquema de la sección 3.3.2. Suelen usarse en aplicaciones que tienen como principal objetivo la recolección de datos. La principal tarea del centro de fusión será recoger los datos enviados por cada sensor y, a partir de los datos locales obtenidos, construir una estimación global de la magnitud deseada. Sin embargo, a diferencia del procesado centralizado descrito en la sección 3.2, aquí se imponen ciertas restricciones a la información transmitida por los sensores, de modo que el centro de fusión no va a disponer de toda la información recogida por cada uno de ellos. En este apartado estudiaremos dos propuestas para el procesamiento distribuido con centro de fusión [4].

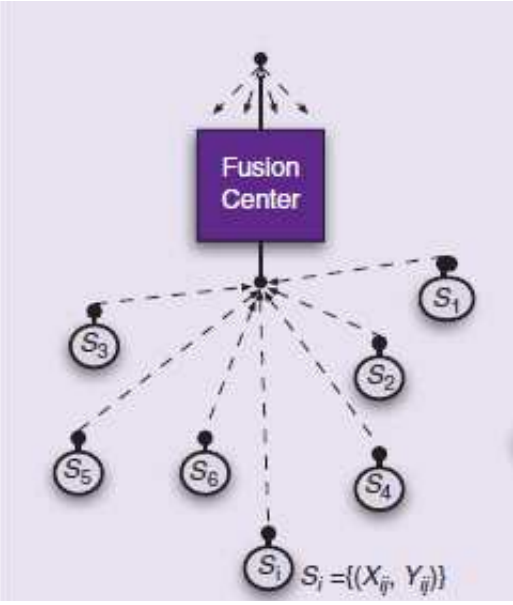


Figura 3.3: Procesado distribuido con centro de fusión [4].

Una primera alternativa consiste en seleccionar una serie de nodos "principales", que recolectarán y comunicarán la información de los datos de un subconjunto de nodos de su entorno al centro de fusión mencionado. En caso de que el número de nodos sea pequeño y los datos enviados correctamente elegidos, podremos llegar a obtener muy buenos re-

sultados y hacer que el coste disminuya. Para ello deberemos dividir la red de sensores en grupos ("clusters"), dentro de los cuales habrá un nodo al que consideraremos principal ("cluster head"), que recuperará los datos de los nodos que pertenecen a su "cluster". Dado que en cada "cluster" los nodos se van a encontrar muy cerca unos de otros, el coste de comunicación va a ser pequeño, porque no habrá mucha distancia que recorrer. Una vez que el nodo principal almacena los datos, los filtrará para saber cuáles son los realmente relevantes, y estos serán los que transmitirá al centro de fusión.

La otra alternativa está basada en una extensión del teorema de Stone [39], cuyo objetivo es determinar la cantidad mínima de información necesaria para que el aprendizaje sea consistente, teniendo en cuenta que cada sensor adquiere un pequeño conjunto de datos de entrenamiento y que la comunicación con el centro de fusión está limitada. Para resolver esta cuestión, suponemos que cada sensor adquiere sólo un conjunto de datos de entrenamiento, $S_i = (X_i, Y_i)$ para $1 \leq i \leq n$. Por otro lado, en el momento que el centro de fusión desea conocer el valor de la magnitud observada en una nueva posición, $\mathbf{x} \in X \subset R$, la difunde a los sensores, solicitando que le envíen su estima, obtenida a partir de la información almacenada en sus propias posiciones. Al mismo tiempo, el ancho de banda de la red limita la respuesta de cada sensor como mucho a 1 bit por petición. Esto es, el sensor decide si responder o no a la petición del centro de fusión, y si decide responder enviará un 1 o un 0 en base a su algoritmo de decisión local. Una vez que el centro de fusión dispone de la respuesta de todos los sensores de la red, combinará dicha información, obteniendo una estima de la magnitud deseada Y. Este esquema se muestra en la Figura 3.4, donde se puede ver que el centro de fusión tiene un canal de difusión hacia los sensores, por el que envía la nueva posición X, mientras que desde cada sensor disponemos de un canal ascendente punto-a-punto inalámbrico por el que podremos enviar como mucho 1 bit por transmisión [4]. En [40]se demuestra que este tipo de esquemas permiten una detección y una estimación universalmente consistente conforme el número de sensores dentro de un área determinada tiende a infinito.

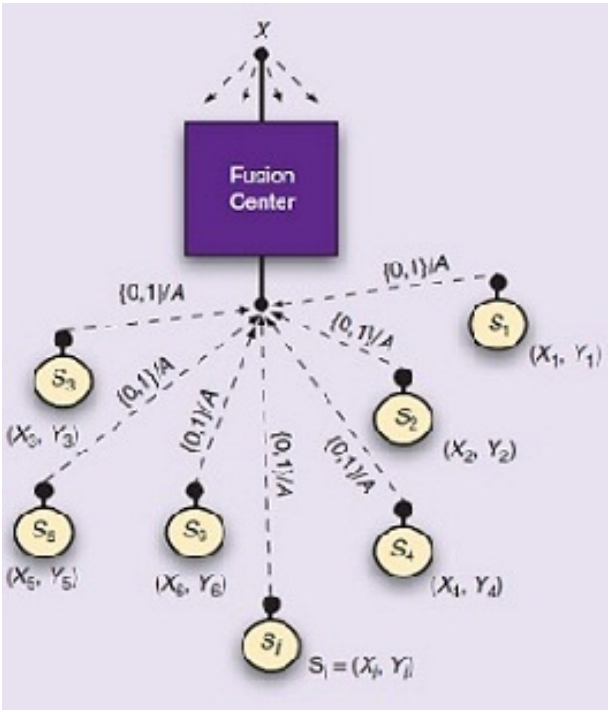


Figura 3.4: Procesamiento distribuido en redes con centro de fusión y con restricciones en el canal ascendente [4].

3.3.2. Procesamiento distribuido con topología "ad-hoc"

En este tipo de redes la topología puede ser dinámica y desconocida a priori. En ellas puede existir un centro de fusión, como en el apartado anterior, pero la gran diferencia es que los sensores trabajan de forma autónoma y toman decisiones de manera independiente. Como se puede ver en la Figura 3.5, existe comunicación local entre los sensores, lo que permite el aprendizaje conjunto. Este cambio de enfoque permite un aprendizaje y un procesamiento interno dentro de la propia red, con los sensores colaborando unos con otros y haciendo que la comunicación sea más eficiente.

Otro aspecto importante de estas redes es que no tienen una infraestructura previa, formada por enlaces inalámbricos fijos, de forma que cualquier estación puede ser origen, destino o enrutador, proporcionando flexibilidad y autonomía a las mismas. Debido a esto, resulta muy difícil utilizar protocolos de encaminamiento convencionales, ya que la

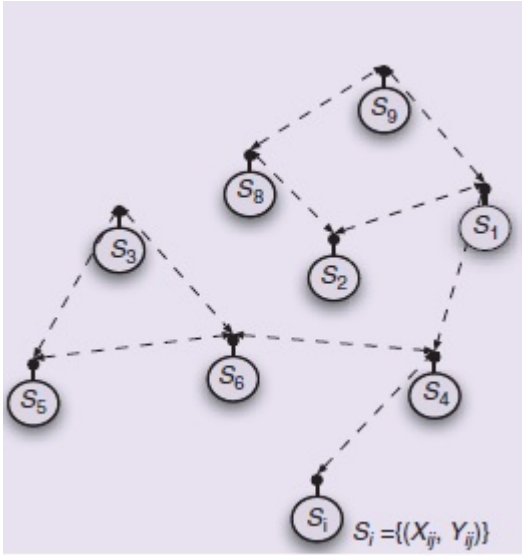


Figura 3.5: Red con procesamiento distribuido [4].

topología de la red puede estar continuamente cambiando. En estas redes, la conexión se realiza típicamente por descubrimiento de nodos, donde un nodo puede comunicarse con otros que estén fuera de su alcance conectándose a través de nodos cercanos que sí estén conectados directamente a la red deseada. Podemos decir entonces que la conexión se realiza habitualmente a través de múltiples saltos ("multi-hop routing") [2].

A pesar de las ventajas que nos pueden proporcionar las redes "ad hoc" convencionales (como pueden ser el tipo de redes desplegadas bajo tecnología inalámbrica Bluetooth o WiFi [41]), necesitamos una serie de características añadidas, que provienen de algoritmos y protocolos necesarios para la implementación de las WSNs con topología "ad hoc", que estas redes no pueden proporcionarnos y que debemos incorporar [2]. Entre las principales características a mencionar de esta clase de redes tenemos:

- El número de nodos en las WSNs puede ser varios órdenes de magnitud superior al número típico de nodos de las redes "ad hoc" convencionales.

- Los nodos sensores en las WSNs están densamente distribuidos y su movilidad es habitualmente reducida, mientras que en una red "ad hoc" convencional los nodos pueden estar en continuo movimiento.

- Los nodos sensores en las WSNs están basados en comunicación por difusión, mientras que las redes "ad hoc" convencionales se basan en comunicación punto a punto y los nodos son mucho más fiables.

- Los nodos sensores en las WSNs están mucho más limitados en potencia, capacidad de cálculo y memoria, que en las redes "ad-hoc" convencionales.

- Los nodos sensores en las WSNs no pueden tener identificación global, debido a su elevado número, su sencillez y el tipo de distribución que tienen.

- Los nodos de las WSNs están densamente distribuidos, es decir, los nodos vecinos se encuentran muy cerca. Por este motivo, se utiliza la comunicación a través de múltiples saltos ("multi-hop routing"), ya que se espera que el consumo de energía sea mucho menor. Además, esto implica que los niveles de potencia de transmisión sean bajos y que los efectos negativos sobre la señal, como atenuación, dispersión o difracción, sean menores que en las comunicaciones inalámbricas de medias/largas distancias.

3.4. Procesamiento centralizado vs. distribuido

Para elegir entre ambos tipos de procesado debemos estudiar las ventajas e inconvenientes de cada uno. Comenzaremos describiendo las desventajas de usar una red distribuida frente a una centralizada [18]:

- **Encaminamiento.** Es una tarea complicada, porque los terminales se pueden mover o fallar. Un camino óptimo en un momento dado puede convertirse en un camino inexistente pocos segundos después. Por lo tanto, vamos a necesitar la ayuda de protocolos de encaminamiento adaptativos [42].

- **Seguridad.** Es difícil tener la certeza de que el otro sensor es quien dice ser. Se puede resolver con ayuda de la criptografía simétrica, donde los nodos de la red utilizan una clave secreta de grupo para tareas de autentificación y para generar las claves de cifrado [43].

- **Software.** Deberá ser más complejo, ya que tendremos que manejar información proveniente de varios usuarios. A la hora de implementar e implantar el software necesario surgen una serie de preguntas que no son fáciles de responder: qué lenguaje de programación debe usarse, qué aplicaciones, cuántos sensores son necesarios, qué trabajo deberían realizar los sensores, etc.

A pesar de las desventajas, el procesamiento distribuido tiene más ventajas que inconvenientes, tal y como se muestra a continuación [2], [4], [18]:

- **Capacidad de cálculo.** La capacidad de cálculo será mayor con menor coste, ya que es un trabajo conjunto de todos los sensores que forman la red.
- **Comunicación.** Los sensores pueden colocarse a distancias cortas, de forma que la potencia necesaria para llevar a cabo la comunicación será menor que la de enviar los datos directamente al centro de fusión.
- **Configurabilidad.** Una vez que el despliegue de los nodos ha terminado, estos no requieren intervención humana, sino que la configuración y el mantenimiento de los mismos se realiza de manera autónoma. Estos nodos serán programables y se adaptarán a los cambios de la red.
- **Crecimiento.** En el momento en que la carga de procesamiento de la red aumente de forma considerable o queramos aumentar la potencia de la red, sólo tendremos que añadir más sensores a la misma sin tener que modificar nuestra estación base o centro de fusión. En el caso de que la carga de la red disminuyera, también podríamos reducir la capacidad de la red fácilmente sin coste alguno.
- **Fiabilidad.** En teoría será mayor, ya que la información recogida está dividida entre un gran número de sensores. Por lo tanto, si uno falla no implica que no podremos conocer el valor de la magnitud deseada en el área en el que estemos trabajando. Podríamos decir que los datos están duplicados, de modo que seríamos capaces de inferir el valor deseado a partir de los datos de otro sensor. En este caso, los datos no van a depender de la disponibilidad de una sola fuente, sino que tendremos varias fuentes de las cuales extraer la información.

- **Recursos.** Existe un mejor aprovechamiento de los recursos de los que disponemos.

- **Topología arbitraria.** Los nodos se despliegan de manera aleatoria, es decir, se conectan entre sí de manera dinámica, haciendo que la red adopte múltiples formas y puedan descubrirse o mantenerse rutas.

- **Precisión.** Es bien conocido que el uso de un gran número de sensores baratos y poco fiables puede proporcionar resultados más precisos que el uso de unos pocos sensores caros y más fiables [44].

Por otro lado, dentro del procesado distribuido hemos hablado de dos topologías: "ad-hoc" y con centro de fusión. La topología "ad-hoc" volverá a tener más ventajas que la topología centralizada con centro de fusión, ya que este último tiene muchas similitudes con el procesamiento centralizado, compartiendo algunas desventajas con el mismo como las relacionadas con el aprovechamiento de los recursos, la fiabilidad y la comunicación.

3.5. Discusión

En este capítulo hemos estudiado dos tipos de procesamiento de datos: centralizado y distribuido. A continuación vamos a discutir cuál sería el más adecuado para nuestro problema. Para ello debemos tener en cuenta que queremos trabajar con una red de sensores inalámbricos, que están distribuidos de manera aleatoria, y que deseamos que el coste en todos los sentidos (potencia, consumo, coste, etc.) sea el menor posible. Atendiendo a estas características, podemos decir que la topología de red que más nos interesa es la "ad-hoc" por diversas razones:

1. No existe un centro de fusión que reciba todos los datos, ni que sea el encargado de procesar toda la información, algo que podría resultar muy caro en el caso de que dicho sistema falle. Es preferible contar con dispositivos más pequeños y que tengan capacidad de procesamiento, como son los sensores. En caso de querer aumentar la

potencia o la capacidad de procesamiento, resulta más barato añadir nuevos sensores a la red que instalar otro sistema central de procesamiento.

2. Establecer comunicación con sensores vecinos será más barato que tener que comunicar dichos sensores con el centro de fusión, que puede encontrarse bastante alejado en comparación con nuestros sensores vecinos.

3. Tendremos mayor fiabilidad y seguridad, porque el hecho de que falle un sensor no implicará la caída total de la red, mientras que con el procesamiento centralizado o el centro de fusión sí que perderíamos toda la información en caso de fallo del centro de fusión.

4. No vamos a necesitar trabajar con anchos de banda muy elevados, ya que la cantidad de información enviada será reducida, lo que implicará utilizar un menor ancho de banda.

5. Estas redes se adaptan de manera autónoma a los cambios de la red.

En consecuencia, en lo sucesivo nos centraremos en redes de sensores con topología "ad hoc" sin centro de fusión y procesamiento distribuido, que será realizado directamente por los sensores de la propia red.

Capítulo 4

Algoritmos de procesado centralizado y distribuido

4.1. Introducción

Una vez que hemos analizado cuál es el tipo de procesado apropiado para nuestro caso de estudio, vamos a examinar en más profundidad los métodos que nos ayudarán a llevarlo a cabo. Sin embargo, antes de exponer los distintos algoritmos de procesado centralizado y distribuido, vamos a describir brevemente las diferencias entre los modelos paramétricos y no paramétricos. Esto es debido a que algunos de los algoritmos que vamos a presentar en este capítulo suelen utilizarse como *estimadores no paramétricos*, pero en este Proyecto Final de Carrera vamos a realizar una adaptación de los mismos, para poder utilizarlos como *estimadores paramétricos*.

Por un lado, tenemos que tanto unos como otros realizan una fase de entrenamiento, donde intentan estimar una serie de parámetros e hiperparámetros (modelos paramétricos y no paramétricos, respectivamente) de una función determinada en base a unos datos de entrenamiento, con el objetivo de predecir el valor de dicha función en cualquier punto del conjunto de test usando los parámetros estimados en la fase anterior. Por otro lado, podemos decir que entre sus principales diferencias tenemos las que se enuncian a continuación:

- Un modelo paramétrico sólo necesita los parámetros estimados para predecir el valor de la función en la fase de test. Por el contrario, un modelo no paramétrico necesita los hiperparámetros estimados y, además, los datos de entrenamiento.

- En el caso de *modelos estadísticos*, un modelo paramétrico se basa en especificar una función de distribución concreta parametrizada, mientras que un modelo no paramétrico no se basa en una función de distribución particular, sino en una familia genérica que contiene la información general conocida de los datos.

Por tanto, podríamos decir que el modelo no paramétrico es más robusto, ya que no hace suposiciones muy restrictivas sobre la forma de la función de distribución de los datos, que podrían llevar a resultados pésimos en caso de no acertar, pero más costoso computacionalmente, especialmente en la fase de test, debido a que un aumento de los datos de entrenamiento podría implicar un modelo muy complejo.

Como hemos explicado en el Capítulo 3, la topología de red que vamos a considerar en nuestro problema es "ad-hoc". En la Figura 4.1 podemos ver una de las redes con las que hemos trabajado. En nuestra investigación no hemos considerado el hecho de dividir la red en "clusters", sino que la comunicación entre los sensores se hará de forma secuencial. Es decir, deberemos establecer una ruta con la que podamos visitar todos los sensores de la red una sola vez. Este problema es equivalente al problema del viajante (TSP, "Traveling Salesman Problem"), tal y como explicaremos en el Capítulo 5, aunque ahora deba resolverse de un modo distribuido en lugar de centralizado.

La finalidad de este capítulo será resolver un problema de optimización de una función de coste cuadrática, donde asumiremos que cada sensor nos proporcionará un único dato de entrenamiento. Para ello, vamos a considerar una red de N sensores distribuidos aleatoriamente sobre un área determinada. Nuestra función a minimizar, $C_f(\theta)$, vendrá definida como la suma de una función de coste cuadrática, $J_f(\theta)$, y un término de regularización, $R_f(\theta)$, como se puede ver en las siguientes expresiones:

$$C_f(\theta) = J_f(\theta) + R_f(\theta), \tag{4.1}$$

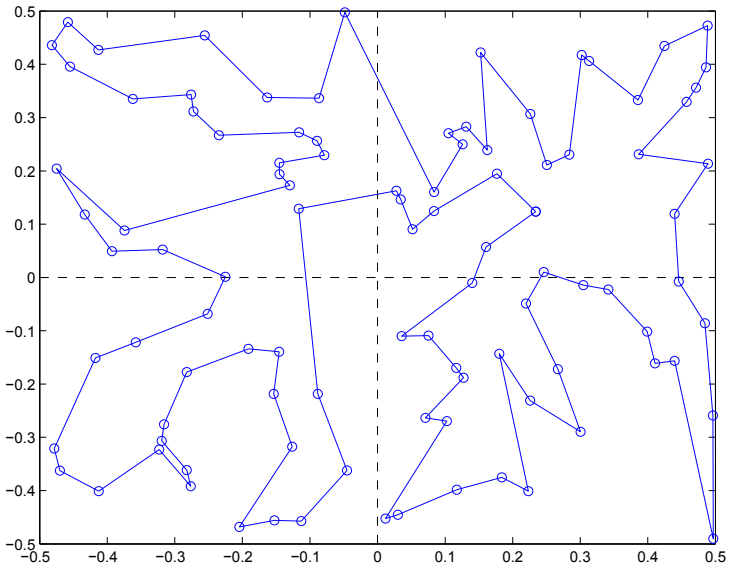


Figura 4.1: Ejemplo de una red utilizada en las simulaciones: 100 sensores distribuidos aleatoriamente conectados por un camino único.

siendo

$$J_f(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left(f(\mathbf{x}_n;\theta) - y_n\right)^2 \tag{4.2}$$

la función de coste cuadrática, donde $f(\mathbf{x}_n;\theta)$ es la función que describe la relación entre la magnitud medida y la posición de cada sensor, $\mathbf{x}_n$, y que dependerá de un vector de parámetros θ, e y_n es la medida aportada por cada sensor, mientras que

$$R_f(\theta) = \lambda \|\theta\|_2^2 \tag{4.3}$$

es el término de regularización, que penaliza los valores elevados del vector de parámetros, siendo λ un parámetro que controla el peso otorgado al término de regularización en relación con la función de coste.

A partir de la función a minimizar, $C_f(\theta)$, podremos conseguir la estima de los parámetros resolviendo el siguiente problema de optimización:

$$\hat{\theta} = \arg\min_{\theta} C_f(\theta) = \arg\min_{\theta}\left\{\frac{1}{N}\sum_{n=1}^{N}\left(f(\mathbf{x}_n;\theta) - y_n\right)^2 + \lambda\|\theta\|_2^2\right\}. \tag{4.4}$$

El resto del capítulo lo dedicaremos a describir una serie de métodos para resolver nuestro problema de optimización, tanto centralizada como distribuida. Con el fin de cumplir este propósito, en primer lugar analizaremos una serie de algoritmos de optimización centralizada, basados en el descenso de gradiente, que nos ayudarán posteriormente en la resolución de los problemas de optimización distribuida. Estos algoritmos distribuidos se diferenciarán, principalmente, en la manera de intercambio de mensajes por parte de los sensores, llevando a cabo todos ellos los pasos que se describen a continuación:

1. **Generación de las medidas**. Esto se debe a que, a la hora de realizar las simulaciones no contamos con una red de sensores real, de modo que hemos tenido que generar unas medidas de manera sintética con ayuda de MATLAB. En el Capítulo 6 explicaremos con más detalle cómo hemos llevado a cabo esta tarea.

2. **Creación de una ruta.** Para realizar las actualizaciones de cada iteración, con cualquiera de los algoritmos, deberemos recorrer una ruta que visite todos los sensores una sola vez, tratando de minimizar la distancia media de los saltos entre los distintos nodos. Existen varias maneras de resolver este problema, como veremos en el Capítulo 5.

3. **Optimización de los parámetros.** Esta será la tarea más importante y donde veremos las principales diferencias. El propósito de este capítulo es revisar en profundidad las diversas opciones existentes en este apartado.

El contenido de este capítulo está dividio en tres secciones, en las que intentaremos explicar los algoritmos de procesado centralizado y distribuido que utilizaremos para llegar a cumplir nuestros objetivos.

- En la Sección 4.2 describiremos los tres algoritmos de procesado centralizado usados en este Proyecto Final de Carrera, y que nos ayudarán en la optimización de los parámetros de los métodos distribuidos: el descenso de gradiente, el gradiente conjugado y el gradiente conjugado escalado.

- En la Sección 4.3 analizaremos el primero de los dos métodos distribuidos considerados: el algoritmo del Subgradiente.
- En la Sección 4.4 estudiaremos el segundo método distribuido analizado: el algoritmo de Proyecciones Alternas.

4.2. Optimización Centralizada

El estudio de la optimización de los parámetros se ha basado en tres algoritmos centralizados: el descenso de gradiente, el gradiente conjugado y el gradiente conjugado escalado. A continuación se describe brevemente cada uno de ellos.

4.2.1. Algoritmo de descenso de gradiente

El método del descenso de gradiente es una extensión del *método de Laplace*, que se aplica a la estima asintótica de integrales de funciones analíticas. En general, los matemáticos han atribuido el desarrollo del algoritmo de descenso de gradiente al físico Peter Debye, que en 1909 trabajó en un estudio asintótico de funciones de Bessel, y que reconoció haber tomado prestada la idea de un artículo de Bernhard Riemann de 1863 [45].

El descenso de gradiente es un método iterativo [46], cuya finalidad es optimizar un vector de pesos, que serán nuestros parámetros. Consideramos una función a minimizar $C_f(\theta)$, donde θ son los parámetros a estimar, que establece una relación entre el espacio de parámetros y los números reales, esto es $C_f(\theta) : \Theta \rightarrow R$. Nuestro propósito será encontrar una solución óptima, a la que llamaremos $\hat{\theta}_{opt}$, que deberá satisfacer la siguiente condición:

$$C_f\left(\hat{\theta}_{opt}\right) \leq C_f\left(\hat{\theta}\right) \ \forall\, \theta \in \Theta. \tag{4.5}$$

Para ello partimos de un vector inicial, al que llamaremos $\hat{\theta}(0)$, a partir del cual generaremos una secuencia de vectores de pesos, de forma que la función a minimizar, $C_f\left(\hat{\theta}\right)$, vaya disminuyendo en cada iteración. Dado que el gradiente de la función de coste en un punto del espacio de parámetros, $\nabla C_f\left(\hat{\theta}\right)$, nos indica la dirección de máxima variación

local de la misma, es fácil ver que, para llegar al valor óptimo de θ, tendremos que movernos en la dirección opuesta al gradiente. En consecuencia, la ecuación de actualización del vector de parámetros en la iteración k-ésima resulta

$$\hat{\theta}\left(k+1\right) = \hat{\theta}\left(k\right) - \frac{1}{2}\mu_k \nabla C_f\left(\hat{\theta}(k)\right), \tag{4.6}$$

donde μ_k es un vector de parámetros de ajuste, esto es, un vector de constantes positivas típicamente $0 < \mu_k < 1$, que controla la convergencia del algoritmo: cuanto mayor sea μ_k más rápido convergerá, pero un valor de μ_k demasiado alto podría provovar su divergencia.

Para cada iteración, la dirección de minimización se elige en función del signo del gradiente, de tal manera que siempre nos moveremos en la dirección contraria al mismo. Realizando la elección óptima de μ_k en cada iteración se consigue que los gradientes consecutivos sean ortogonales, esto es, que

$$\nabla C_f\left(\hat{\theta}\left(k+1\right)\right) \nabla C_f\left(\hat{\theta}\left(k\right)\right)^{\top} = 0, \tag{4.7}$$

lo que implica que la búsqueda se haga en zig-zag [47].

En resumen, podemos decir que los pasos para implementar este método son los que se muestran en el Algoritmo 4.1, donde K es el número total de iteraciones necesarias para obtener la convergencia. En él se muestra la dependencia, ya comentada, de dicho algoritmo con la elección del parámetro μ más adecuado para que pueda cumplirse (4.5). Aunque la solución óptima implica usar un parámetro μ variable con cada iteración k, una alternativa más sencilla y utilizada en la práctica consiste en elegir un valor fijo de μ suficientemente pequeño [46].

A pesar de la sencillez de este algoritmo, presenta una serie de inconvenientes importantes, tales como la gran dependencia del éxito del mismo en función de la elección de los parámetros (μ_k, ϵ y K), su tendencia a tomar como solución óptima mínimos locales de la función, y su baja tasa de convergencia (esto es, su lentitud). Por esta razón, si se quiere mejorar la velocidad de convergencia se suele recurrir al algoritmo de gradiente conjugado, que estudiaremos en el siguiente apartado.

1. Seleccionar un punto inicial $\theta = \hat{\theta}(0)$.

2. Para k = 1,....,K:

   2.1 Escoger una dirección de descenso:

   $$d_k = -\nabla C_f\left(\hat{\theta}(k)\right).$$

   2.2 Realizar una búsqueda lineal que seleccione un paso μ_k tal que:

   $$C_f\left(\hat{\theta}(k+1)\right) < C_f\left(\hat{\theta}(k)\right).$$

   2.3 Actualizar el vector de pesos:

   $$\hat{\theta}(k+1) = \hat{\theta}(k) - \tfrac{1}{2}\mu_k \nabla C_f\left(\hat{\theta}(k)\right).$$

   2.4 Hacer un test de convergencia. Por ejemplo, verificar si la diferencia entre el vector de pesos de la iteración actual y la anterior es menor que un cierto valor ϵ o si se ha alcanzado un número máximo de iteraciones. Si se cumple dicha condición, el algoritmo habrá convergido y se finalizará la búsqueda. En caso contrario, se continúa con la iteración siguiente.

Algoritmo 4.1: Pseudocódigo para llevar a cabo la implementación del algoritmo de descenso de gradiente.

4.2.2. Algoritmo de gradiente conjugado

La historia del método de gradiente conjugado lineal comenzó con las investigaciones de Cornelius Lanczos, Magnus Hestenes y otros matemáticos en el Instituto para Análisis Numérico de Los Ángeles (EE.UU), así como con las investigaciones independientes de Eduard Stiefel en el Instituto Federal de Tecnología de Zurich (Suiza). Este algoritmo fue publicado en 1952 en un artículo conjunto de Hestenes y Stiefel como un método iterativo capaz de resolver sistemas lineales con matrices de coeficientes positivos [48]. Por otro lado, alrededor de 1960 Fletcher y Reeves desarrollaron el método de gradiente conjugado no lineal [49]. Estos métodos pertenecen a una clase de algoritmos de optimización sin restricciones, que están caracterizados por el uso de poca memoria, ya que no necesitan

almacenar la matriz de coeficientes, y por buena convergencia tanto local como global. Cuentan con múltiples técnicas para resolver sistemas de ecuaciones lineales complejos, además de poder adaptarse correctamente para solucionar problemas de optimización no lineales [50].

A continuación vamos a facilitar una breve explicación de qué hace exactamente este método y cómo lo hace. Al igual que en el algoritmo del descenso de gradiente, visto en el apartado anterior, consideramos una función a minimizar $C_f(\theta)$, donde θ son los parámetros a estimar, que establece una relación entre el espacio de parámetros y los números reales, esto es $C_f(\theta) : \Theta \rightarrow R$. De nuevo, nuestro objetivo será encontrar el valor óptimo de dichos parámetros, $\hat{\theta}_{opt}$, que hagan cumplir (4.5) vista en la sección 4.2.1. La principal diferencia con el algoritmo de descenso de gradiente está en el modo de elegir la dirección de descenso [50]. Conocido un punto inicial, $\hat{\theta}(0) \in \Theta$ y un conjunto de direcciones, $[\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_k]$, el vector de parámetros en la iteración $(k+1)$-ésima se obtiene como

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \alpha_k \mathbf{p}_k, \tag{4.8}$$

donde $\mathbf{p}_k$, nos indica la dirección de búsqueda, que inicialmente será la opuesta al gradiente en $\hat{\theta}(0)$:

$$\mathbf{p}_0 = -\nabla C_f\left(\hat{\theta}(0)\right) = -\mathbf{g}_0. \tag{4.9}$$

Posteriormente, una vez pasada la primera iteración, $\mathbf{p}_k$ se calculará como:

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1}, \tag{4.10}$$

siendo $\mathbf{g_k}$ el gradiente de la función a minimizar evaluado en el vector de parámetros estimado tras la iteración k-ésima:

$$\mathbf{g}_k = \nabla C_f\left(\hat{\theta}(k)\right), \tag{4.11}$$

que nos indica la dirección en la que debemos movernos localmente para encontrar el mínimo de la función, y

$$\beta_k = \frac{\left\|\nabla C_f\left(\hat{\theta}\left(k\right)\right)\right\|^2}{\left\|\nabla C_f\left(\hat{\theta}\left(k-1\right)\right)\right\|^2} \tag{4.12}$$

el parámetro que nos ayuda a actualizar el vector de direcciones $\mathbf{p}_k$, ponderando la contribución del gradiente actual y los valores pasados. Por último, α_k se obtiene tratando de minimizar la función $C_f\left(\hat{\theta}\left(k+1\right)\right)$, tal y como se hace para el parámetro μ_k en el descenso de gradiente. Nótese que si fijáramos $\beta_k = 0$ y $\alpha_k = \frac{1}{2}\mu_k$ obtendríamos el algoritmo de descenso de gradiente, de modo que el gradiente conjugado podríamos verlo como una generalización del primero. Por otro lado, fijando $\beta_k = 1$ estaríamos dando igual peso a todos los gradientes, haciendo que el algoritmo nunca olvide sus valores pasados.

Para implementar este algoritmo deberemos seguir los pasos mostrados en el Algoritmo 4.2, donde K es el número total de iteraciones necesarias para obtener la convergencia.

1. Seleccionar un punto inicial $\theta = \hat{\theta}\left(0\right)$.

2. Para k=1,....,K:

   2.1 Escoger una dirección de búsqueda, $\mathbf{p}_k$, de acuerdo con (4.9) y (4.10).

   2.2 Calcular el parámetro de actualización de $\mathbf{p}_k$, β_k, usando (4.12), escogiendo el tamaño del paso α_k de tal modo que:

$$C_f\left(\hat{\theta}\left(k+1\right)\right) < C_f\left(\hat{\theta}\left(k\right)\right) \tag{4.13}$$

   2.3 Actualizar el vector de pesos como se mostró en (4.8).

   2.4 Hacer un test de convergencia. Por ejemplo, verificar si la diferencia entre el vector de pesos de la iteración actual y la anterior es menor que un cierto valor ϵ o si se ha alcanzado un número máximo de iteraciones. Si se cumple dicha condición, el algoritmo habrá convergido y se finalizará la búsqueda. En caso contrario, se continúa con la iteración siguiente.

Algoritmo 4.2: Pseudocódigo para llevar a cabo la implementación del algoritmo de gradiente conjugado.

Como mencionamos en la sección anterior, la principal ventaja de este algoritmo con respecto al descenso de gradiente es que la velocidad de convergencia aumenta. Sin embargo, este algoritmo no resuelve el problema de la elección de parámetros, ya que hay parámetros que aún deben ser elegidos por el usuario. Este problema se intenta resolver mediante el algoritmo del gradiente conjugado escalado, que se explica en el siguiente apartado.

4.2.3. Algoritmo de gradiente conjugado escalado

La existencia de este método se debe a que los algoritmos basados en descenso de gradiente tienen una tasa de convergencia baja y, además, el éxito de los mismos es en gran parte dependiente de los parámetros elegidos por el usuario, como puede ser μ en el descenso de gradiente, o α en el gradiente conjugado [51]. La novedad de este método, con respecto a los vistos anteriormente, es que evita la búsqueda en línea usando el método de Levenberg-Marquardt para escalar el tamaño del paso [52].

Al igual que en los dos algoritmos anteriores, consideraremos una función a minimizar $C_f(\theta)$, donde θ son los parámetros a estimar, que establece una relación entre el espacio de parámetros y los números reales, esto es $C_f(\theta) : \Theta \rightarrow R$. Por otro lado, como vimos en el apartado anterior, tendremos una secuencia de k vectores distintos de $\mathbf{0}$ $[\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_k]$, que nos indicarán la dirección de búsqueda, y la secuencia de pesos se generará como en la Ecuación (4.8). Sin embargo, en este método el valor de α_k se obtendrá a partir de la siguiente expresión:

$$\alpha_k = \frac{\mu_k}{\delta_k}, \tag{4.14}$$

donde

$$\mu_k = -\mathbf{p}_k^\top \nabla C_f(\hat{\theta}(k)) \tag{4.15}$$

y

$$\delta_k = \mathbf{p}_k^\top \nabla^2 C_f(\hat{\theta}(k))\, \mathbf{p}_k. \tag{4.16}$$

Desafortunadamente, al realizar las primeras pruebas con este algortimo se demostró que convergía a puntos no estacionarios de la función de coste. Esto se debía a que el algoritmo trabajaba con matrices definidas positivas y las aproximaciones cuadráticas podían ser muy malas si la estima actual del vector de parámetros se encontraba muy lejos del mínimo deseado [53]. Para solucionarlo se introdujo un escalar λ_k en (4.16), de forma que δ_k finalmente resulta:

$$\delta_k \;=\; \mathbf{p}_k^{\top}\, \nabla^2 C_f\left(\hat{\theta}\,(k)\right)\; \mathbf{p}_k \;+\; \lambda_k\; |\mathbf{p}_k|^2\,. \tag{4.17}$$

Si el parámetro λ_k es grande, el tamaño del paso será pequeño, dado que α_k es inversamente proporcional al valor de δ_k, de acuerdo con (4.14). En cambio, si λ_k es pequeño, el tamaño del paso será grande. Por lo tanto, podemos decir que λ_k escala el tamaño del paso α_k de una forma artificial. Esto hace que las aproximaciones no siempre sean buenas [51]. Para conseguir una buena aproximación, se define un nuevo parámetro,

$$\Delta_k \;=\; \frac{2\;\delta_k \left[C_f\left(\hat{\theta}\,(k)\right) - C_f\left(\hat{\theta}\,(k+1)\right)\right]}{\mu_k^2}, \tag{4.18}$$

como mecanismo para aumentar/decrementar el parámetro λ_k, de forma que:

$$\lambda_{k+1} \;=\; \begin{cases} 4\lambda_k, & \text{si } \Delta_k < 0,25; \\ \lambda_k, & \text{si } 0,25 \leq \Delta_k \leq 0,75; \\ \frac{\lambda_k}{2}, & \text{si } \Delta_k > 0,75 \end{cases}$$

Para implementar este método deberemos seguir los pasos mostrados en el Algoritmo 4.3, donde K es el número total de iteraciones necesarias para obtener la convergencia. Nótese que este algoritmo utiliza una evaluación extra para la estimación de segundo orden de la función de error en la dirección de búsqueda, e introduce un método de Levenberg-Marquardt para el escalado del tamaño del desplazamiento, α_k. Las principales ventajas con respecto a los dos algoritmos estudiados en apartados anteriores son el aumento de la velocidad de convergencia y la independencia del éxito del algoritmo con respecto a los parámetros elegidos por el usuario. Sin embargo, estas dos ventajas implican mayor complejidad a la hora de implementar este algoritmo.

1. Seleccionar un punto inicial $\theta = \hat{\theta}(0)$.

2. Para k=1,....,K:

   2.1 Escoger una dirección de búsqueda, $\mathbf{p}_k$, de acuerdo con (4.9) y (4.10).

   2.2 Calcular el parámetro de actualización, β_k, de $\mathbf{p}_k$ como en (4.12) y calcular el tamaño del paso α_k de acuerdo con (4.14).

   2.3 Actualizar el vector de pesos siguiendo (4.8).

   2.4 Hacer un test de convergencia. Por ejemplo, verificar si la diferencia entre el vector de pesos de la iteración actual y la anterior es menor que un cierto valor ϵ o si se ha alcanzado un número máximo de iteraciones. Si se cumple dicha condición, el algoritmo habrá convergido y se finalizará la búsqueda. En caso contrario, se continúa con la iteración siguiente.

Algoritmo 4.3: Pseudocódigo para llevar a cabo la implementación del algoritmo de gradiente conjugado escalado.

4.3. Optimización Distribuida: Algoritmo del Subgradiente

Los métodos de subgradiente fueron desarrollados por Naum Z. Shor y otros matemáticos entre 1960 y 1970 [54]. Son algoritmos iterativos, cuyo objetivo es minimizar una función convexa, que está formada por la suma de un determinado número de componentes. Esta aproximación, que veremos a continuación, ha sido muy útil para resolver problemas de optimización de mínimos cuadrados, obteniéndose tasas de convergencia mejores que las proporcionadas por los métodos de descenso de gradiente [4]. La finalidad es la misma que en los casos anteriores: queremos conocer el valor de unos parámetros partiendo de otros parámetros iniciales, escogidos aleatoriamente. La diferencia es que ahora no vamos a suponer la existencia de un nodo central que conoce todas las observaciones, sino que estas están distribuidas en los diferentes nodos de la red. Afortunadamente, la función a

minimizar original, dada por (4.4), puede reescribirse como:

$$C_f(\theta) = \sum_{n=1}^{N} \left[J_f^{(n)}(\theta) + R_f^{(n)}(\theta) \right], \tag{4.19}$$

donde

$$J_f^{(n)}(\theta) = \frac{1}{N} \left(f(\mathbf{x}_n; \theta) - y_n \right)^2 \tag{4.20}$$

es el componente de la función de coste correspondiente a la observación del sensor n-ésimo, y

$$R_f^{(n)}(\theta) = \lambda_n \|\theta\|_2^2, \tag{4.21}$$

con

$$\sum_{n=1}^{N} \lambda_n = \lambda, \tag{4.22}$$

es la parte del término de regularización asociada al sensor n-ésimo. Por lo tanto, el problema de optimización de los parámetros se puede reformular como sigue:

$$\begin{aligned} \hat{\theta} &= \arg \min_{\theta} \left\{ \sum_{n=1}^{N} \left[J_f^{(n)}(\theta) + R_f^{(n)}(\theta) \right] \right\} \\ &= \arg \min_{\theta} \left\{ \frac{1}{N} \sum_{n=1}^{N} \left(f(\mathbf{x}_n; \theta) - y_n \right)^2 + \sum_{n=1}^{N} \lambda_n \|\theta\|_2^2 \right\} \end{aligned} \tag{4.23}$$

El algoritmo del subgradiente, en vez de actualizar θ utilizando la información de todos los sensores a la vez, como hacen los métodos de gradiente, realiza dicha actualización usando los datos de cada sensor por separado. El proceso de estimación distribuida que estudiamos se realizará en dos pasos, tal y como se muestra en el Algoritmo 4.4. Nótese que ahora hay un doble bucle, que está en función de las iteraciones necesarias para converger y el número de nodos de la red. Por tanto, el parámetro $\mu_{k,n}$ dependerá de ambas variables, aunque en la práctica será constante para cada iteración.

En la Figura 4.2 se muestra un ejemplo de las iteraciones de este método. El primer sensor tendrá un vector de parámetros iniciales $\hat{\theta}(0)$, con el que empezaremos a ejecutar

1. Establecer un camino para recorrer la red de manera ordenada en el que visitemos todos los sensores una sola vez, tal y como se muestra en el Capítulo 5.

2. Realizar las actualizaciones iterativamente a medida que se va visitando cada sensor.

   2.1 Seleccionar un punto inicial $\hat{\theta}(0) = \hat{\theta}(0, N)$.

   2.2 Para $k = 0, ..., K-1$:

$$\hat{\theta}(k+1, 0) = \hat{\theta}(k, N). \tag{4.24}$$

   Para $n = 1, ..., N$:

$$\hat{\theta}(k+1, n) = \hat{\theta}(k+1, n-1) - \mu_{k,n} \nabla C_f^{(n)} \left(\hat{\theta}(k+1, n-1) \right). \tag{4.25}$$

   end

   Hacer test de convergencia. Por ejemplo, verificar si la diferencia entre el vector de pesos de la iteración actual y la anterior es menor que un cierto valor ϵ o si se ha alcanzado un número máximo de iteraciones. Si se cumple dicha condición, el algoritmo habrá convergido y se finalizará la búsqueda. En caso contrario, se continúa con la siguiente iteración.

Algoritmo 4.4: Pseudocódigo para llevar a cabo la implementación del algoritmo del subgradiente, donde K es el número de iteraciones necesarias para que el algoritmo converja y N es el número de sensores de la red.

el algoritmo, obteniendo una primera estimación de los parámetros, $\hat{\theta}(1, 1)$, que dependerá únicamente de sus propios datos de entrenamiento. Una vez realizada esta primera estimación, transmitimos dichos parámetros al segundo sensor de la lista, que llevará a cabo el mismo proceso de estimación, obteniendo $\hat{\theta}(1, 2)$, enviándoselo al tercer sensor de la lista y así sucesivamente. Debemos destacar que no se envían los datos, sino las estimas calculadas en cada iteración, $\hat{\theta}(k, n)$. Esto es muy importante, ya que, si se utilizan modelos sencillos con un número reducido de parámetros, no necesitaremos sistemas con

gran capacidad de transmisión ni de almacenamiento y la energía consumida será menor.

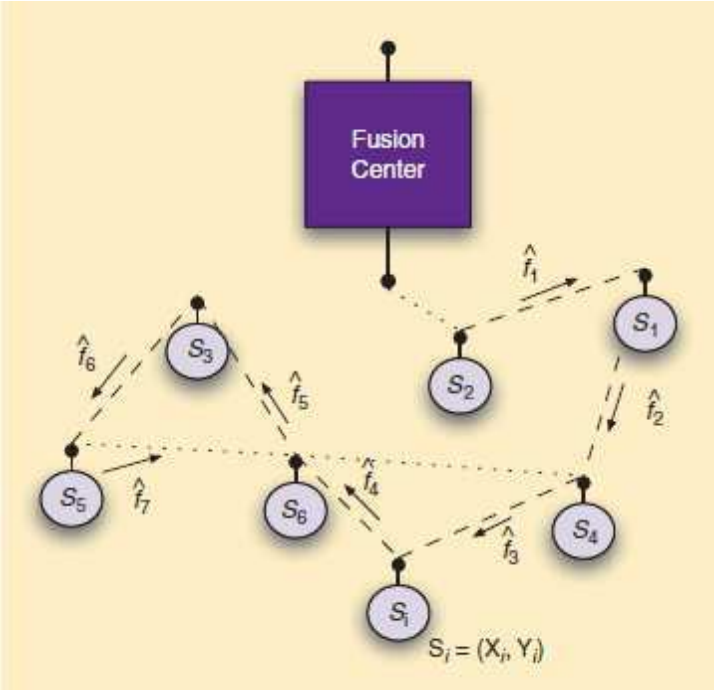


Figura 4.2: Ejemplo de entrenamiento distribuido con el método del subgradiente [4].

4.4. Optimización Distribuida: Algoritmo de proyecciones alternas

El primero en estudiar el problema de las proyecciones alternas fue John Von Neumann en 1933, considerando el problema de encontrar una intersección entre dos subespacios cerrados, C *y* D, y proponiendo una solución mediante la proyección sobre ellos, alternando de forma iterativa [55]. Como se ve en la Figura 4.3, la idea era partir de un punto inicial x_1 e ir proyectando sobre cada hiperplano alternativamente, de forma que esto generara una sucesión que convergiera hacia la intersección de los hiperplanos, $C \cap D$.

Nuestro objetivo será, de nuevo, resolver la Ecuación (4.4) vista en la sección 4.1, utilizando una formulación distribuida similar a la de (4.23). Para construir el algoritmo, asumiremos que el sensor n podrá pedir los datos almacenados por cualquier nodo $j \in$

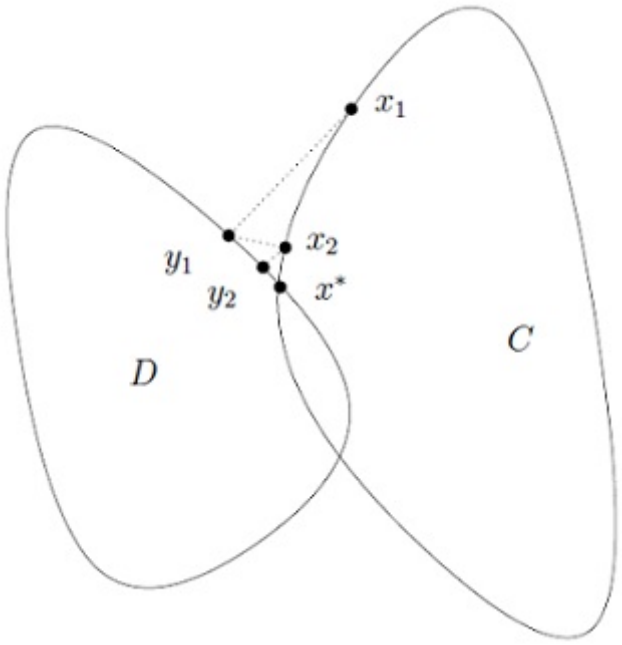


Figura 4.3: Un ejemplo de las primeras iteraciones del algoritmo de Proyecciones Alternas. Ambas secuencias convergen en el punto x* $\in$ $C \cap D$ [5].

N_n, donde N_n representa el conjunto de sensores vecinos de n, incluyendo el propio sensor n. De esta forma, obtendremos una estima global de los parámetros a partir de las estimas locales, aunque con una restricción, en la que se impone que el valor estimado por cada uno de los sensores vecinos j, calculado en función de la posición del sensor n y los parámetros estimados por ellos mismos, sea igual al valor estimado por el sensor n en su propia posición. Matemáticamente:

$$f_j\left(\mathbf{x}_n;\theta_j\right) \;=\; f_n\left(\mathbf{x}_n;\theta_n\right) \quad \forall j \;\in\; N_n,\; \forall\, n \;\in \{1,....,N\}$$

Teniendo en cuenta esto, la función a minimizar para este algoritmo viene dada por 4.19, siendo el componente n-ésimo de la función de coste

$$C_f^{(n)} = \sum_{j\in N_n} \left(f_n\left(\mathbf{x}_n;\hat{\theta}_n\right) - z_{jn}\right)^2, \tag{4.26}$$

con

$$z_{jn} \;=\; \begin{cases} y_n, & \text{si } j = n; \\ f_j\left(\mathbf{x}_n;\theta_j\right), & \text{si } j \neq n; \end{cases}$$

y el término de regularización n-ésimo,

$$R_f^{(n)}(\theta) = \lambda_n \|\theta_n\|_2^2 . \tag{4.27}$$

Por lo tanto, el problema de optimización se puede reformular a nivel local como se muestra en la siguiente expresión:

$$\hat{\theta}_n = \arg\min_{\theta} \left\{ C_f^{(n)} + R_f^{(n)} \right\} . \tag{4.28}$$

El proceso de estimación distribuida que estudiamos se realizará en tres pasos, como veremos en el Algoritmo 4.5, donde z_{jn} es la magnitud medida/estimada por el sensor j en la posición del sensor n, T es el número de iteraciones que necesita el algoritmo de proyecciones alternas para converger, K es el número de iteraciones que el algoritmo de optimización usado en cada proyección (descenso de gradiente, gradiente conjugado o gradiente conjugado escalado) necesita para converger y N es el número de sensores de la red. Cada sensor n almacenará la variable auxiliar $z_n = f_n\left(\mathbf{x}_n; \hat{\theta}_n\right) \in R$, que será interpretada como la estima realizada por el sensor n en su propia posición $\mathbf{x}_n \; \forall \; n \in \{1, ..., N\}$. Esta variable en la primera iteración se corresponderá con el valor medido por el sensor. Una vez que todos los sensores han almacenado su medida, el algoritmo empezará a llevar a cabo operaciones locales en orden secuencial. Esto implica que necesitaremos la ayuda de alguno de los métodos del Problema del Viajante, que estudiaremos en el siguiente capítulo. Cada sensor n mandará un mensaje a sus vecinos, pidiéndoles la estima que cada sensor vecino j tiene sobre su magnitud, $z_{jn} = f_j\left(\mathbf{x}_n; \hat{\theta}_j\right)$. Este mensaje se corresponde con lo que el sensor j cree que debería valer y en la posición del sensor n, $\mathbf{x}_n$. Con estas variables el algoritmo calculará la solución a (4.28), usando como datos de entrenamiento $\{(\mathbf{x}_n, z_{jn}) \; j \in N_n, \; \forall \; n \in \{1, ..., N\}\}$. En este punto, necesitaremos hacer uso de los métodos de optimización vistos en la sección 4.2. Una vez estimados los parámetros correspondientes, el sensor n actualizará sus mensajes variables, $z_{nj} = f_n\left(\mathbf{x}_j; \hat{\theta}_n\right)$, asociados a cada sensor vecino j. El significado de z_{nj} es el del valor de la magnitud que el sensor n estima que tiene el sensor vecino j, en su posición correspondiente $\mathbf{x}_j$, teniendo en cuenta los parámetros estimados por él mismo, $\hat{\theta}_n$. Como los sensores van actualizando y enviando los mensajes variables z_{nj} a sus vecinos, podemos deducir que el algoritmo

permite que la información local se propague globalmente si la red está completamente conectada.

1. Obtener los vecinos de cada nodo sensor como:

$$N_n = \left\{ j \in \{1, ..., N\} : \|\mathbf{x}_n - \mathbf{x}_j\|_2 \leq r \right\} \qquad \forall\, n \in \{1, ..., N\}\,, \tag{4.29}$$

donde r es una cierta distancia prefijada ($r > 0$) y $n \in N_n$, ya que $\|\mathbf{x}_n - \mathbf{x}_j\|_2 = 0 < r$.

2. Establecer un camino para recorrer la red de manera ordenada con el que visitemos todos los sensores una sola vez, tal y como se muestra en el Capítulo 5.

3. Realizar las actualizaciones iterativamente a medida que se va visitando cada sensor:

   3.1 Para t=1,....,T:

   Para n=1,....,N:

$$\forall j \in N_n : z_{jn,t-1} = f_{j,t-1}\left(\mathbf{x}_n; \hat{\theta}_{j,t-1}\right) \tag{4.30}$$

   Para k=0,....,K-1

$$\hat{\theta}_{n,t}\,(k+1) = \hat{\theta}_{n,t}\,(k) - \frac{1}{2}\mu_{n,t}(k)\nabla C_f^{(n)}\left(\hat{\theta}_{n,t}\,(k)\right) \tag{4.31}$$

   end

   Actualizar las estimas para los vecinos:

$$\forall j \in N_n : z_{nj,t} = f_{n,t}\left(\mathbf{x}_j; \hat{\theta}_{n,t}(K)\right) \tag{4.32}$$

   end

Algoritmo 4.5: Pseudocódigo para llevar a cabo la implementación del algoritmo de proyecciones alternas.

En la Figura 4.4 se puede ver un ejemplo del entrenamiento de este algoritmo. En

ella, el nodo 1 pide a sus nodos vecinos, 4 y 7, que le envíen lo que ellos estiman que debería valer y en $\mathbf{x}_1$, es decir, el valor que los nodos vecinos han calculado teniendo en cuenta la posición del sensor 1 y los parámetros estimados por los sensores 4 y 7 en la iteración anterior ($z_{7,t-1}$ y $z_{4,t-1}$). Una vez que el sensor 1 tiene las estimas de sus vecinos y la suya propia, estima unos nuevos parámetros y transmite a sus nodos vecinos la nueva estimación de la magnitud a medir, calculada teniendo en cuenta la posición de sus vecinos y los parámetros que acaba de estimar. De esta forma, envía $z_{4,t}$ y $z_{7,t}$ a 4 y 7, respectivamente. Por otro lado, se observa que el sensor 5 está realizando la misma operación con sus correspondientes nodos vecinos (3 y 6).

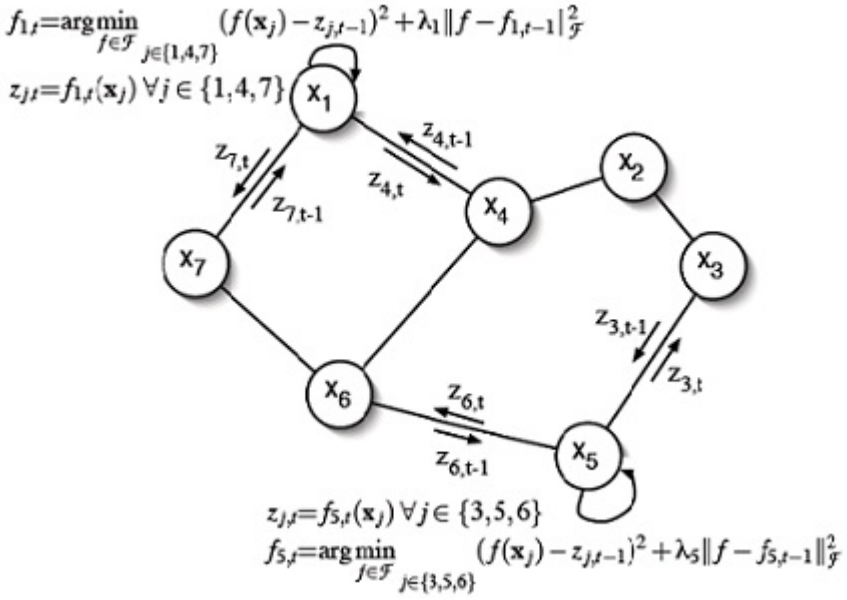


Figura 4.4: Ejemplo del entrenamiento del algoritmo de proyecciones alternas [6].

4.5. Discusión

En este capítulo hemos trabajado bajo el siguiente escenario: una red de N sensores distribuidos aleatoriamente sobre un área determinada, donde cada uno nos proporciona un único dato de entrenamiento (medida). El objetivo es estudiar una serie de aproximaciones para resolver el problema de optimización de una función de coste cuadrática.

Antes de estudiar los algoritmos de estimación distribuida, hemos revisado tres algoritmos de procesado centralizado sobre los datos: el descenso de gradiente, el gradiente conjugado y el gradiente conjugado escalado. El descenso de gradiente es el más sencillo de todos, pero es el que presenta mayores desventajas, tales como su baja velocidad de convergencia, que puede mejorarse con los algoritmos restantes, la dependencia del éxito del mismo con la elección de los parámetros por parte de los usuarios (por ejemplo, el parámetro μ), característica que sólo mejora con el gradiente conjugado escalado, y su tendencia a tomar como solución final mínimos locales de la función, propiedad que no soluciona ninguno de los algoritmos mencionados.

Una vez analizados los métodos de optimización centralizada, hemos pasado a estudiar dos algoritmos de procesado distribuido, que permiten llevar a cabo la optimización de manera eficiente en una WSN, donde es muy costoso transmitir todas las medidas a un nodo central:

- **Subgradiente.** La optimización en este método se consigue actualizando los parámetros usando los datos de cada sensor de forma independiente. Esto es, un sensor reajusta los parámetros globales en base a los que el sensor anterior a él en la ruta le envía y a la información que dicho sensor tiene almacenado.

- **Proyecciones alternas.** Este método se basa en realizar la optimización de los parámetros de cada sensor con ayuda de las estimas de la magnitud deseada calculadas por sus sensores vecinos. En este caso la actualización de los parámetros en el sensor n se hace a partir de las estimas que los sensores vecinos j tienen sobre la medida del sensor n y la suya propia.

En ambos casos, conseguimos llevar a cabo una estimación global teniendo en cuenta las estimas locales. Sin embargo, una de las principales diferencias entre el método del subgradiente y el algoritmo de proyecciones alternas es que este último envía a sus nodos vecinos las estimas calculadas de la magnitud deseada y no los parámetros, lo que supone un ahorro en el consumo de potencia y capacidad de almacenamiento cuando se trabaja con modelos no paramétricos o modelos paramétricos con un elevado número de parámetros.

Capítulo 5

Problema del viajante

5.1. Introducción

Como ya hemos comentado en los capítulos anteriores, para poder aplicar los algoritmos de procesado distribuido necesitamos recorrer la red de sensores de una forma secuencial y con el menor coste posible. Por lo tanto, el problema que aquí se nos plantea es encontrar la ruta óptima que debe seguir la información transmitida entre los sensores. Por ruta óptima consideramos aquella que nos proporcione el camino de menor longitud con el que, partiendo de un sensor que puede ser elegido arbitrariamente, visitemos todos los sensores una sola vez y volvamos al punto de partida. Este problema es conocido en teoría de grafos como el problema de encontrar un ciclo Hamiltoniano óptimo, y se sabe que se trata de un problema NP-completo ("Nondeterministic Polynomial-time complete") [56]. Es decir, no existe un algoritmo capaz de resolverlo de manera óptima en un tiempo polinómico, sino que es una función exponencial del tamaño del grafo.

La primera referencia a este problema fue encontrada por Müller-Merbach en un manual de 1832 [57], donde se definía el problema y se mostraban algunos ejemplos de rutas seguidas por comerciantes, que debían pasar por una serie de ciudades y regresar a su ciudad de origen intentando minimizar la distancia total recorrida, pero no existía ningún tipo de tratamiento matemático sobre el mismo. Por otro lado, W. Hamilton y T. Kirkman fueron los primeros en desarrollar matemáticamente este problema alrededor de 1856

[58]. Sin embargo, hasta 1930 en las Universidades de Viena y Harvard no se realizó una formulación general del problema de manera independiente, que fue llevada a cabo especialmente por Karl Menger. El interés por la resolución de este problema ha ido creciendo, y actualmente es uno de los más estudiados en el campo de la optimización combinatoria computacional, ya que existen numerosas aplicaciones prácticas en las que aparecen diversas variantes del mismo [59].

El contenido de este capítulo se va dividir en tres secciones, en las que se discutirá formalmente el problema del viajante y a continuación estudiaremos dos de los métodos capaces de resolver el problema que acabamos de plantear de manera distribuida.

- En la Sección 5.2 definiremos el problema del viajante de manera más formal, introduciendo las restricciones consideradas y la notación matemática.
- En la Sección 5.3 describiremos el funcionamiento del primer algoritmo: vecino más próximo basado en la búsqueda del nodo más cercano.
- En la Sección 5.4 explicaremos el funcionamiento de un algoritmo más sofisticado: el método de inserción basado en la búsqueda del más cercano al que añadiremos una modificación con el fin de permitir su implementación distribuida con un coste computacional razonable.

5.2. Descripción del problema del viajante

Formalmente podemos enunciar el problema del viajante como sigue. Se dispone de un grafo $G = (N, A)$, donde $\mathcal{N}$ indica el conjunto de nodos de la red y A el conjunto de vértices (esto es, conexiones entre nodos) de la misma, siendo $(a_1, ..., a_n)$ un camino válido en G si se tiene que $(a_i, a_{i+1}) \in A \; \forall \; i \in \{1, ..., n\}$. Decimos que G contiene un *circuito Hamiltoniano* si existe un camino $(a_1, ..., a_{n+1})$ en G tal que:

- n es el número de nodos en $\mathcal{N}$.
- $a_i \neq a_j \; \forall \, i, j \in \{1, ..., n\}$ tal que $i \neq j$.

- $a_1 = a_{n+1}$.

Por tanto, un circuito Hamiltoniano en G será aquel que pasa exactamente una sola vez por cada nodo i $\in \mathcal{N}$, y tendrá un coste asociado igual a la suma de los costes correspondientes a los vértices que lo constituyen. En este Proyecto Final de Carrera, el coste será la distancia física existente entre dichos nodos, $d(i,j)$, que, como toda métrica, debe cumplir una serie de condiciones [60]:

- **Simetría:** $d(i,j) = d(j,i) \; \forall \; i,j \; \in \; \mathcal{N}$.
- **No negatividad:** $d(i,j) \geq 0 \; \forall \; i,j \; \in \; \mathcal{N}$.
- **Desigualdad triangular:** $d(i,j) + d(j,k) \geq d(i,k) \; \forall \; i,j \; \in \mathcal{N}$.
- **Identidad de nodos indiscernibles:** $d(i,j) \; = \; 0 \; \Leftrightarrow \; i \; = \; j$.

El objetivo será encontrar aquel circuito que nos proporcione la mínima distancia. Resolver este problema de manera exacta requiere explorar todos los caminos posibles utilizando un algoritmo de fuerza bruta o búsqueda exhaustiva, lo cual puede llegar a ser inviable para grafos de tamaño mediano o grande. En consecuencia, se suele recurrir a métodos heurísticos, que se pueden agrupar en dos categorías [61]:

- **Constructivos:** construyen la solución paso a paso, es decir, se van añadiendo elementos a la solución a medida que se van teniendo en cuenta los criterios mencionados. Dentro de ellos nos encontramos con métodos como el vecino más próximo, los métodos de inserción y la heurística de Christofides.
- **De mejora:** métodos que se basan en procedimientos iterativos, que parten de una solución, la modifican y obtienen soluciones parecidas, que pueden ser mejores. Dentro de esta categoría nos encontramos con métodos como el de búsqueda local.

Estos métodos pueden ser programados de manera que operen típicamente en un tiempo proporcional a N^2, donde N es el número de nodos del grafo (sensores de la red), de forma que se podrán implementar en un tiempo razonable. Además, para su implementación únicamente se va a requerir la comunicación entre nodos vecinos, de modo que también van a ser eficientes desde un punto de vista energético.

5.3. Algoritmo del vecino más próximo

El algoritmo del vecino más próximo fue uno de los primeros usados para dar una solución al problema del viajante. Se basa en una idea similar a la del descenso de gradiente: si un sensor sólo conoce sus vecinos más cercanos y debe establecer un camino óptimo, éste empezará con un enlace al vecino más cercano, lo cual constituye el camino óptimo localmente, al igual que el gradiente nos indica la dirección óptima de descenso localmente. En este caso, la ruta a seguir se construye como se muestra en el Algoritmo 5.1.

1. Elegir un nodo aleatoriamente.
2. Buscar el nodo más cercano, que aún no esté en la ruta, y añadirlo a la lista conectándolo con el último añadido.
3. Conectar el primer y último sensor de la lista, una vez se han insertado todos los nodos de los que se dispone.

Algoritmo 5.1: Enumeración de los pasos a realizar en el algoritmo del vecino más próximo.

En la Figura 5.1 se muestra una red de 6 sensores, distribuidos aleatoriamente en un área de $100 \times 100\ m^2$, que nos servirá de ejemplo a lo largo de este capítulo. Cabe destacar que en esta, así como en el resto de figuras que se mostrarán a partir de ahora, dicho área estará normalizada entre -0.5 y 0.5 para ambas coordenadas. En la Figura 5.2 se muestra el camino óptimo para esta red de sensores (obtenido mediante búsqueda exhaustiva), con una distancia aproximada igual a 2.5260.

A continuación vamos a utilizar la red de la Figura 5.1 con el fin de mostrar la creación de la ruta según el método del vecino más próximo. En la Figura 5.3 se observan los pasos que debemos seguir para poder generar la ruta más corta:

1. Partimos de un sensor (marcado en rojo, sensor 5) que calcula la distancia desde su posición a la de los demás nodos no añadidos a la ruta hasta el momento (líneas punteadas azules), tal y como se muestra en la Figura 5.3(a).

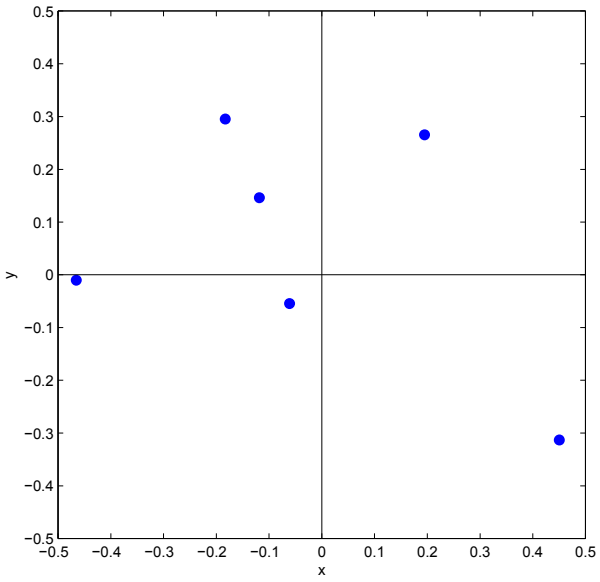


Figura 5.1: Ejemplo de red de 6 sensores distribuidos aleatoriamente en un área con coordenadas normalizadas entre -0.5 y 0.5.

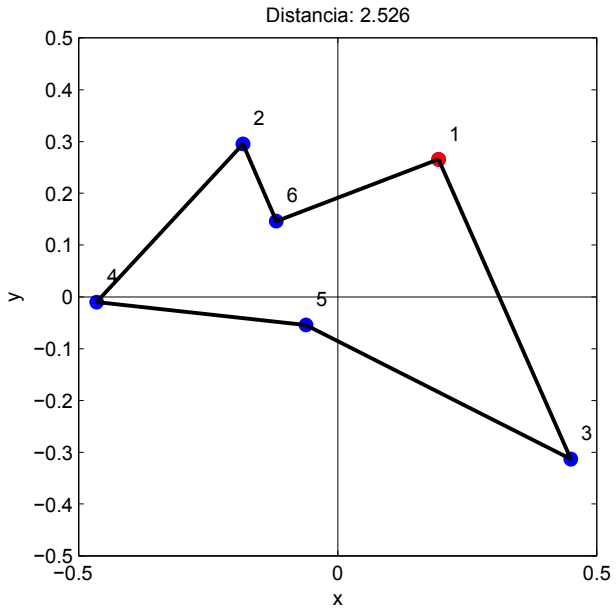


Figura 5.2: Ruta óptima para una red de 6 sensores distribuidos aleatoriamente.

2. En la Figura 5.3(b) vemos que, una vez calculadas esas distancias, se establecerá un enlace con el sensor más cercano (línea negra, sensor 6). Posteriormente se calculará la distancia desde su posición al resto de nodos libres, al igual que hizo el sensor

anteriormente marcado en rojo.

3. A continuación, en la Figura 5.3(c) observamos que, en el momento en que este segundo sensor haya calculado todas las distancias, establecerá igualmente un enlace con el nodo más cercano, y tras esto calculará las distancias al resto de nodos libres.

4. Este proceso se repite para los restantes nodos, tal como se observa en las Figuras 5.3(d) y 5.3(e).

5. Por último, desde el último nodo de la ruta (sensor 4) se establece una conexión con el primero (sensor 5), en la Figura 5.3(f), obteniéndose la ruta final.

A la hora de implementar este algoritmo, con el fin de poder comparar el mayor número de caminos posibles, decidimos calcular dicha ruta iniciando cada iteración con cada uno de los N sensores existentes, de forma que obtuviéramos N rutas posibles, eligiendo finalmente la de menor longitud. En la Figura 5.4 se muestran las 6 rutas obtenidas para la red de 6 sensores de la Figura 5.1, junto con la distancia recorrida, observando que la menor distancia es 2.7536, que en este caso corresponde con la Figura 5.4(e). Como puede verse, no se alcanza en ninguno de los casos mostrados en la Figura 5.4 el camino óptimo de la Figura 5.2, ni siquiera para esta red tan sencilla.

Usando este método se puede demostrar que la relación entre la longitud proporcionada por cada ruta obtenida mediante este algoritmo (d_{NN}) y la óptima (d_{OPT}) es [61]:

$$\frac{d_{NN}\left(G, \mathbf{x}_i\right)}{d_{OPT}\left(G\right)} \leq \frac{1}{2}\left\lceil \log_2(N) \right\rceil + \frac{1}{2}, \tag{5.1}$$

donde $\mathbf{x}_i$ indica la posición del sensor inicial y $\lceil \cdot \rceil$ la parte entera aproximada por arriba. Nótese como el cociente entre la distancia obtenida por el método del vecino más próximo y la del camino óptimo crece logarítmicamente con N, obteniéndose que, por ejemplo para N = 1024, la distancia proporcionada por este método puede ser hasta 5,5 veces la óptima.

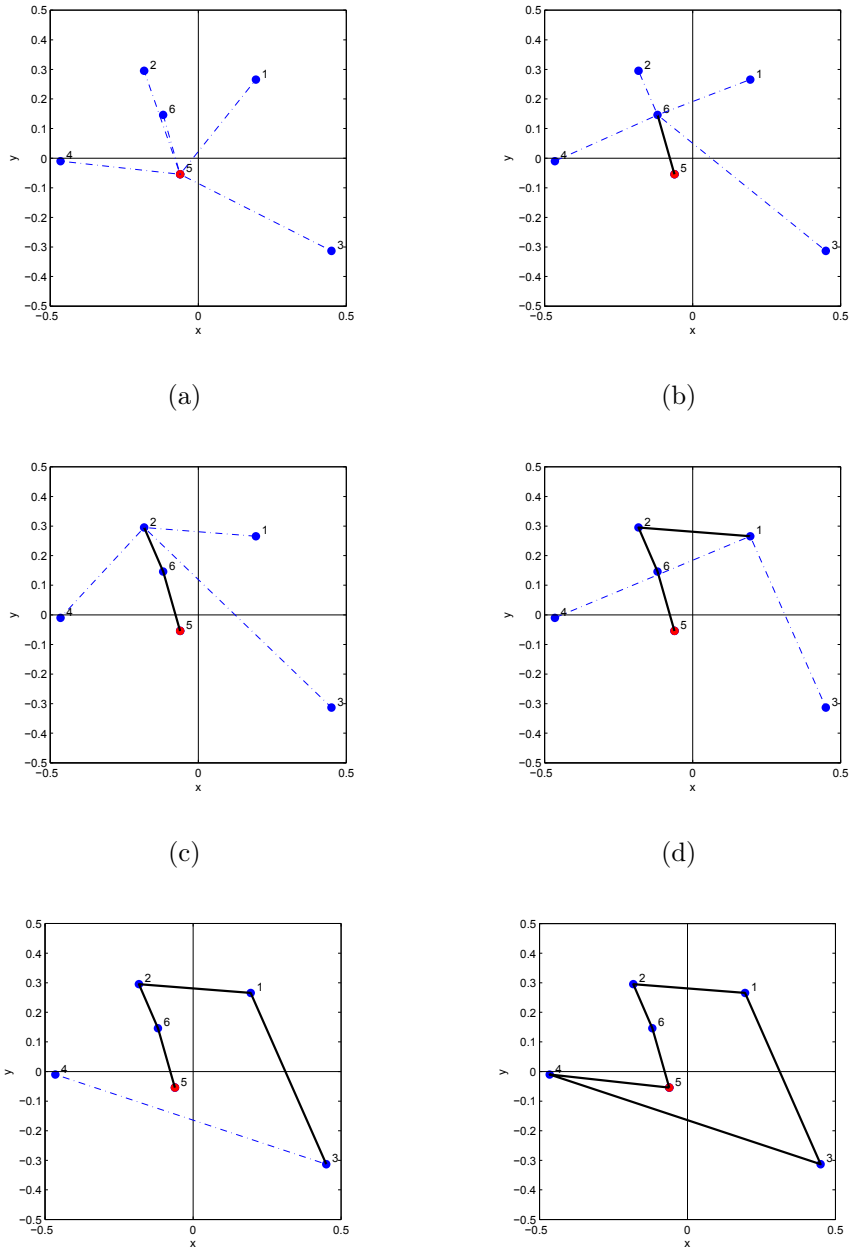


Figura 5.3: Ejemplo de generación de una ruta paso a paso en la red de 6 sensores de la Figura 5.1 utilizando como sensor inicial el marcado en rojo.

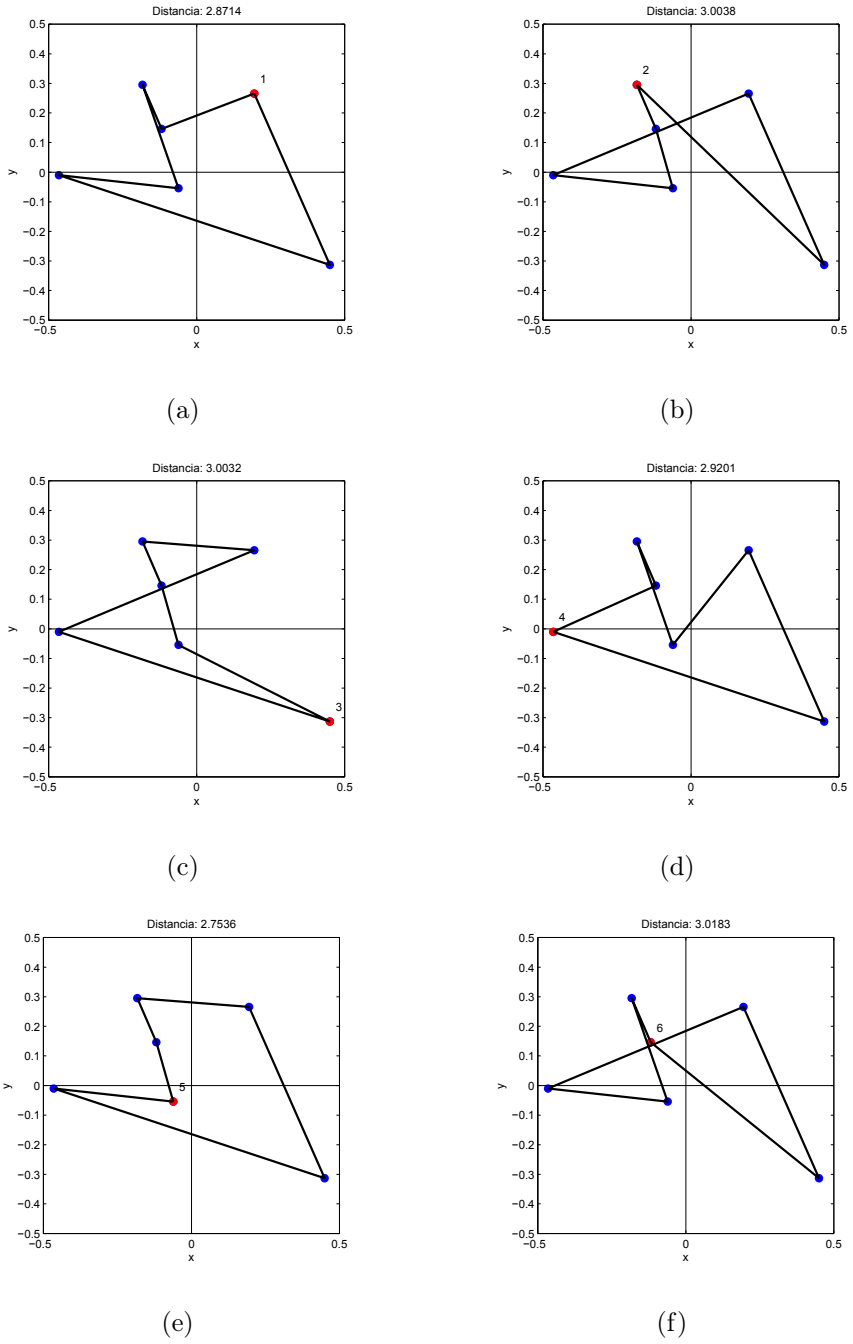


Figura 5.4: Rutas obtenidas aplicando el método del vecino más próximo sobre la red de 6 sensores de la Figura 5.1 partiendo desde cada uno de los 6 sensores de la red. En rojo está marcado el inicio de cada ruta final.

5.4. Métodos de inserción

En este apartado estudiaremos uno de los métodos de inserción más sencillos existente [61], a partir del cual desarrollaremos otra alternativa, en la que intentaremos que el coste computacional disminuya para posibilitar su implementación distribuida. A diferencia del método del vecino más próximo, los métodos de inserción intentan buscar la ruta óptima insertando los nuevos nodos dentro de la ruta. Es decir, el nuevo nodo no tiene que añadirse necesariamente en la última posición de la ruta, sino que podemos insertarlo entre cualquiera de los nodos ya añadidos. El objetivo de esta familia de métodos es construir una aproximación de la ruta óptima de forma progresiva. Sin embargo, antes de describir el método implementado es necesario proporcionar una serie de definiciones que serán importantes para la comprensión del mismo.

- **Definición 1.** Dado un grafo $G = (\mathcal{N}, d)$, donde $\mathcal{N}$ es el conjunto de nodos de la red y d es la distancia entre dichos nodos, y una ruta T formada por un subconjunto $S \subset \mathcal{N}$ de nodos, que llamaremos *subruta* del grafo $(\mathcal{N}, d)$, podemos escribir $a \in T$, es decir $a \in S$, considerando a como una ruta sin costes, dado que contiene un único nodo.

- **Definición 2.** Dado un grafo $G = (\mathcal{N}, d)$, una ruta T y un nodo $k \in \mathcal{N}$, con $k \notin T$, obtendremos la ruta (T, k) como sigue:

  a) Si T contiene un único nodo, a, entonces la ruta de T estaría formada por $(a,\ k,\ a)$.

  b) Si T pasa por más de un nodo, entonces deberemos encontrar un vértice $(i,\ j) \in T$ tal que se minimice:

$$d(i,k) + d(k,j) - d(i,j). \tag{5.2}$$

  En este caso, eliminaríamos la conexión $(i,\ j)$ y añadiríamos las conexiones $(i,\ k)$ y $(k,\ j)$ a la nueva ruta ampliada. Nótese que (5.2) representa la diferen-

cia de longitud entre la ruta actual T y la obtenida reemplazando la conexión $(i,\ j)$ por la conexión $(i,\ k,\ j)$.

Independientemente del caso en el que nos encontremos, deberemos insertar el nodo k dentro de la ruta T.

La clasificación de los métodos de inserción se realiza de acuerdo con distintos criterios: ruta inicial, siguiente salto y posición a insertar. De todos ellos, podemos considerar que el de mayor relevancia es el último y en base al mismo, enunciaremos algunas posibles alternativas que podrían considerarse [61]:

- **Método de inserción más cercano.** El objetivo es insertar el nodo más cercano a cualquiera de los nodos ya insertados en la ruta.
- **Método de inserción más rentable.** El objetivo es encontrar el nodo cuya inserción suponga el menor incremento en la longitud de la ruta.
- **Método de inserción más alejado.** El objetivo es insertar el nodo que se encuentre más alejado de los nodos añadidos a la ruta y, una vez encontrado, localizar el nodo más cercano al mismo, insertándolo en uno de los caminos ya establecidos por dicho nodo.

La gran diferencia de los métodos de inserción con respecto al vecino más próximo es que los resultados obtenidos con estos métodos no proporcionarán una distancia superior al doble de la distancia óptima en ningún caso, con independencia del número de sensores de la red. Considerando d_{INST} la distancia obtenida por estos métodos y d_{OPT} la distancia óptima, la relación existente entre ellas será [61]:

$$\frac{d_{INST}\left(G,\ \mathbf{x_i}\right)}{d_{OPT}\left(G\right)} < 2. \tag{5.3}$$

En las secciones siguientes vamos a explicar el método de inserción elegido, que está basado en la elección del nodo más cercano, y una versión nueva a la que le hemos añadido una modificación, que nos dará ciertas ventajas para la implementación distribuida, a costa de un rendimiento ligeramente inferior.

5.4.1. Método de inserción más cercano.

Dada una ruta T, que inicialmente estará formada por un sensor elegido aleatoriamente, y un nodo $k \in \mathcal{N}$, que será el que deseemos añadir a la ruta, definiremos la distancia $d(T,k)$, entre los nodos que pertenecen a T y un nodo arbitrario k, como la mínima distancia que exista desde el nodo k a cualquier nodo perteneciente a T:

$$d(T,k) = \text{mín}\, d(x,k) \quad \forall\, x \ \in \ T. \tag{5.4}$$

Para construir la ruta deberemos elegir aleatoriamente un nodo de $\mathcal{N}$ para que sea el primero de la lista, al que llamaremos a_1, y que pasará a pertenecer al conjunto T. Una vez elegido $a_1 \ \in \ T$, deberemos calcular la distancia $d(a_1, \mathcal{N}\backslash T)$, es decir, la distancia desde ese primer sensor al resto de nodos que quedan en el grafo y que no pertenecen a T. El nodo que se encuentre a menor distancia será el que incorporemos a T. Una vez añadido este segundo sensor, deberemos calcular, de nuevo, la distancia desde estos dos sensores pertenecientes a T a todos los sensores pertenecientes a $N\backslash T$, volviendo a elegir el de menor distancia según:

$$D_1 = \text{mín} \ \ d(a,k) \quad \forall\, a \ \in T, \ \forall k \ \in \mathcal{N}\backslash T. \tag{5.5}$$

El problema ahora será dónde insertar el nuevo sensor, ya que este no tiene por qué ser el vecino más cercano al último nodo añadido a la lista. Por tanto, deberemos comprobar si es rentable o no romper un enlace (de momento el único existente). Para resolver este dilema deberemos hacer uso de la Definición 2 y (5.6):

$$D_2 = \text{mín}\left\{d(a_i,k) + d(k,a_{i+1}) - d(a_i,a_{i+1})\right\}. \tag{5.6}$$

A continuación, explicaremos los posibles casos que pueden presentarse al insertar un nodo. Nótese que la ruta con la que trabajaremos será $[a\ b\ c]$ y el nodo a insertar será k. Particularizando (5.6) a este ejemplo, la distancia a calcular en cada escenario sería:

$$D_2 = \text{mín}\left\{d(a,k), d(a,k) + d(k,b) - d(a,b), d(b,k) + d(k,c) - d(b,c), d(c,k)\right\}. \tag{5.7}$$

1. **Primer sensor.** Al calcular D_2 se obtiene que la ruta más corta se obtendría insertando el nuevo sensor k delante del primer sensor, de forma que la ruta final sería $[k\ a\ b\ c]$, y D_2 valdría:

$$D_2 = d(a, k). \tag{5.8}$$

2. **Segundo sensor.** Una vez calculado D_2 se llega a que el mínimo incremento en la ruta se produciría insertando el nodo k tras el primer sensor. Es decir, el sensor estaría en la segunda posición de la nueva ruta, desplazando el resto una posición en la misma, obteniéndose la siguiente ruta final $[a\ k\ b\ c]$. Por lo tanto, la expresión (5.6) tendría el siguiente valor:

$$D_2 = d(a, k) + d(k, b) - d(a, b). \tag{5.9}$$

3. **Tercer sensor.** Calculando (5.6) obtenemos que lo más rentable es romper el enlace existente entre b y c, por lo que la ruta final sería $[a\ b\ k\ c]$ y D_2 sería:

$$D_2 = d(b, k) + d(k, c) - d(b, c). \tag{5.10}$$

4. **Último sensor.** En este caso, se insertaría el nodo tras c, obteniendo como ruta final $[a\ b\ c\ k]$ y como resultado de (5.6):

$$D_2 = d(c, k). \tag{5.11}$$

Teniendo en cuenta lo explicado anteriormente, podemos decir que en este método la ruta a seguir se construye como se muestra en el Algoritmo 5.2.

Este método se puede ejecutar en un tiempo proporcional a N^2. El principal problema de este algoritmo es cómo buscar la manera más eficiente de calcular las distancias que aparecen en D_1, que será más costoso a medida que el número de nodos añadidos a T crezca. A modo de ejemplo, vamos a mostrar los primeros pasos necesarios en la pequeña red formada por 5 sensores distribuidos aleatoriamente de la Figura 5.5(a).

1. Elegimos un nodo aleatoriamente. Según la numeración elegida, el primer nodo se corresponde con el nodo en la posición 1 (nodo 1, marcado en rojo en la Figura

1. Elegir un nodo aleatoriamente.

2. Buscar el nodo más cercano a cualquiera de los nodos añadidos a la ruta, calculando:

$$D_1 = \text{mín}\, d(a,k) \quad \forall\, a \ \in T\ \forall k \ \in \mathcal{N}\backslash T. \tag{5.12}$$

3. Insertar el nodo en la lista, calculando:

$$D_2 = \text{mín}\left\{d(a_i,k) + d(k,a_{i+1}) - d(a_i,a_{i+1})\right\}. \tag{5.13}$$

4. Conectar el primer y último sensor de la lista, una vez se han insertado todos los nodos de los que se dispone.

Algoritmo 5.2: Enumeración de los pasos a realizar en el algoritmo del método de inserción del vecino más próximo.

5.5(a)). Desde este nodo calculamos las distancias al resto de nodos y elegimos las dos de menor valor, al igual que hicimos en el método del vecino más próximo. Esas distancias se van a corresponder con la distancia desde el primer nodo (nodo 1) al segundo (nodo 2) y desde el primero (nodo 1) al tercero (nodo 3), d_{1-2} y d_{1-3}, respectivamente (ver Figura 5.5(a)):

$$d_{1-2} = 0{,}2; \quad d_{1-3} = 0{,}82. \tag{5.14}$$

2. Con estas distancias generaremos la siguiente ruta, mostrada en la Figura 5.5(b). Es decir, conectaremos el nodo 1 con el 2, por ser el más cercano al mismo, y el 2 con el 3. En este primer paso, podemos simplificar la Expresión (5.6) y comparar las distancias d_{1-2} y d_{1-3}, ya que las posibles rutas son [1 2 3] o [1 3 2] y sabemos que, al tomar distancias Euclídeas, $d_{2-3} = d_{3-2}$.

3. A partir de este punto, encontraremos las principales diferencias con el método del vecino más próximo, ya que no sólo buscamos el nodo más cercano, sino que debemos insertarlo en la mejor posición de la lista. Por tanto, en este siguiente paso calcularíamos las distancias desde los nodos añadidos a la ruta final (esto es, 1, 2 y

3) hacia el resto de nodos libres (4 y 5), como se puede observar en la Figura 5.5(c).

4. De todas las distancias que se pueden ver en la Figura 5.5(c), elegimos la mínima ($d_{3-5} = 0{,}66$), que se muestra marcada en rojo en la Figura 5.5(d). El nodo que vamos a insertar se corresponde con la posición 5, por tanto lo reconoceremos como nodo 5.

5. El problema que aquí nos encontramos es dónde insertarlo. Para ello, sabemos que la ruta, por el momento, es [1 2 3] y que el nodo 5 está más cerca del 3 (último sensor de la ruta). Podríamos añadirlo delante del sensor 1, entre los sensores 1 y 2, entre 2 y 3 o tras el sensor 3. Para saber cuál de estas opciones debemos elegir, tenemos que aplicar (5.6) usando las distancias calculadas anteriormente y, además, $d_{1-5} = 0{,}89$, $d_{2-5} = 0{,}74$ y $d_{3-5} = 0{,}66$, y calculando

$$\text{mín}\left\{d_{1-5}, d_{1-5} + d_{5-2} - d_{1-2}, d_{2-5} + d_{5-3} - d_{2-3}, d_{3-5}\right\} = \text{mín}\left\{0{,}89,\ 1,42,\ 0{,}57,\ 0{,}66\right\}, \tag{5.15}$$

de modo que, según el criterio elegido, deberíamos romper el enlace 2-3, obteniendo la ruta que se muestra en la Figura 5.5(e): [1 2 5 3]

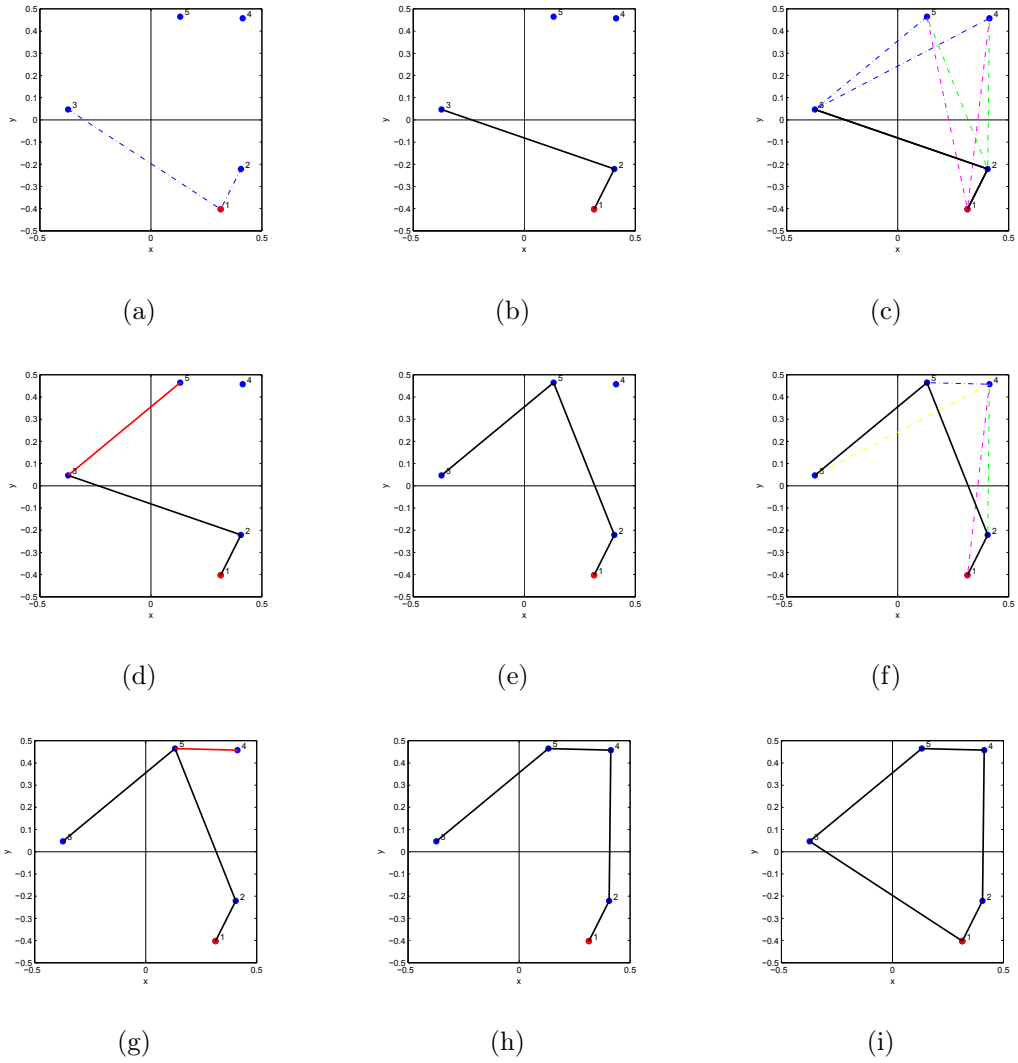
6. De nuevo deberíamos calcular las distancias desde los nodos añadidos a la ruta (1, 2, 3 y 5) hacia el único nodo libre, el 4 (ver Figura 5.5(f)).

7. Volveríamos a elegir la distancia de menor valor e insertaríamos el nodo en la lista, calculando (5.6). Como se puede ver en la Figura 5.5(g), el nodo 4 está más cerca del 5 (distancia=0,28, conexión 4-5). Entonces, las posibilidades serían: [4 1 2 5 3], [1 4 2 5 3], [1 2 4 5 3], [1 2 5 4 3] o [1 2 5 3 4]. En este caso, deberíamos romper el enlace existente ente los nodos 2 y 5, ya que, teniendo en cuenta que $d_{1-5} = 0{,}87$, $d_{2-4} = 0{,}68$ y $d_{3-4} = 0{,}89$, al aplicar (5.6) se obtiene

$$\begin{aligned}&\text{mín}\left\{d_{1-4}, d_{1-4} + d_{4-2} - d_{1-2}, d_{2-4} + d_{4-5} - d_{2-5}, d_{5-4} + d_{4-3} - d_{5-3}, d_{3-4}\right\}\\ &\quad = \text{mín}\left\{0,87, 1{,}34,\ 0{,}22,\ 0{,}51,\ 0{,}89\right\}.\end{aligned} \tag{5.16}$$

La ruta obtenida es [1 2 4 5 3] y se puede ver en la Figura 5.5(h).

8. Finalmente, uniremos el último nodo de la ruta (nodo 3) con el primero (nodo 1), tal y como se muestra en la Figura 5.5(i), obteniendo como ruta final [1 2 4 5 3 1].



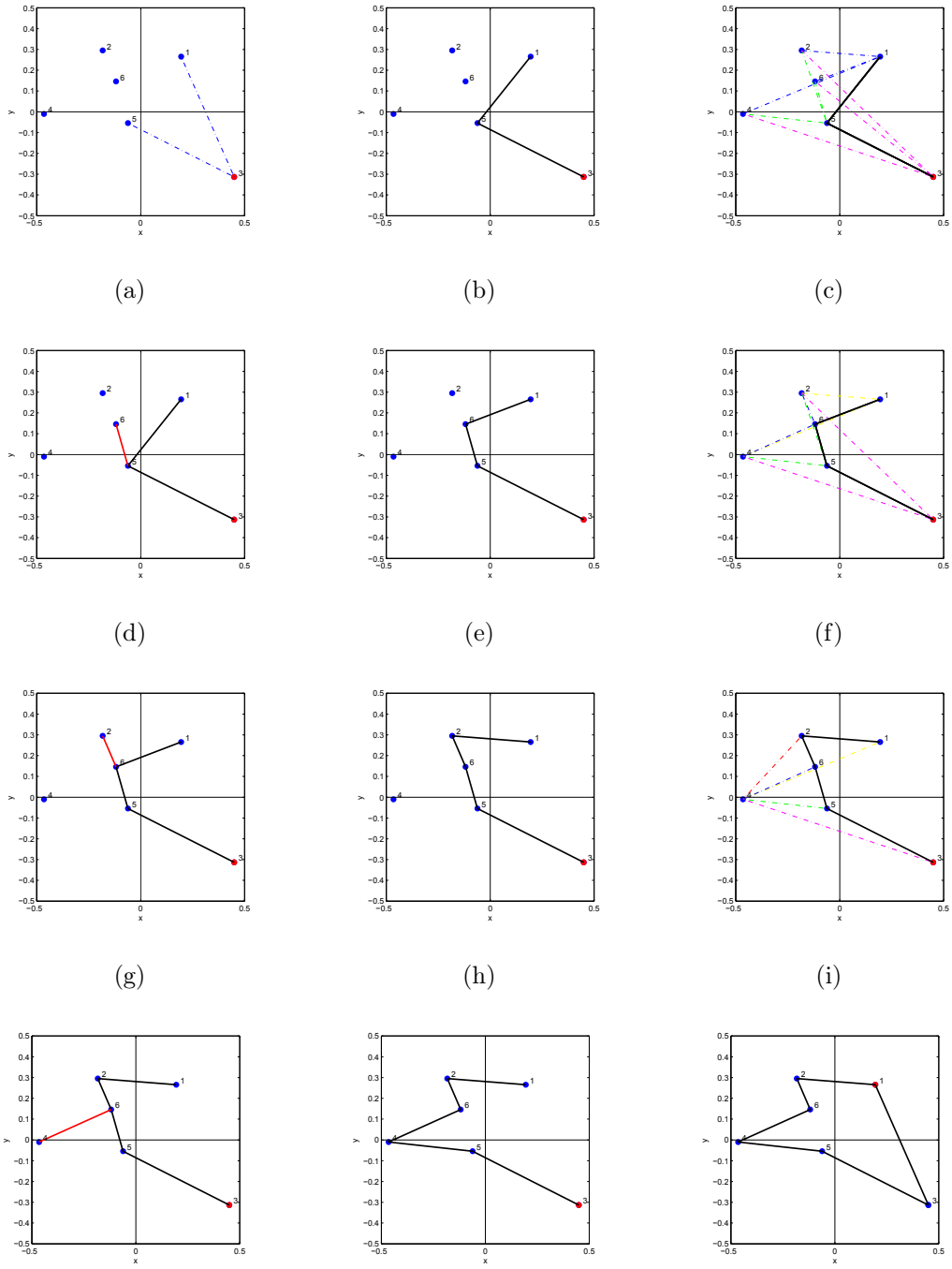
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 5.5: Pasos a seguir al crear una ruta usando el método de inserción más cercano.

Como un segundo ejemplo, en la Figura 5.6 se muestra la generación de la ruta para la red de 6 sensores de la Figura 5.1, mostrándose los pasos seguidos, tal y como hemos explicado para el ejemplo de la red de 5 sensores.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 5.6: Creación de la ruta óptima en la red de 6 sensores distribuidos aleatoriamente de la Figura 5.1 usando el método de inserción más cercano.

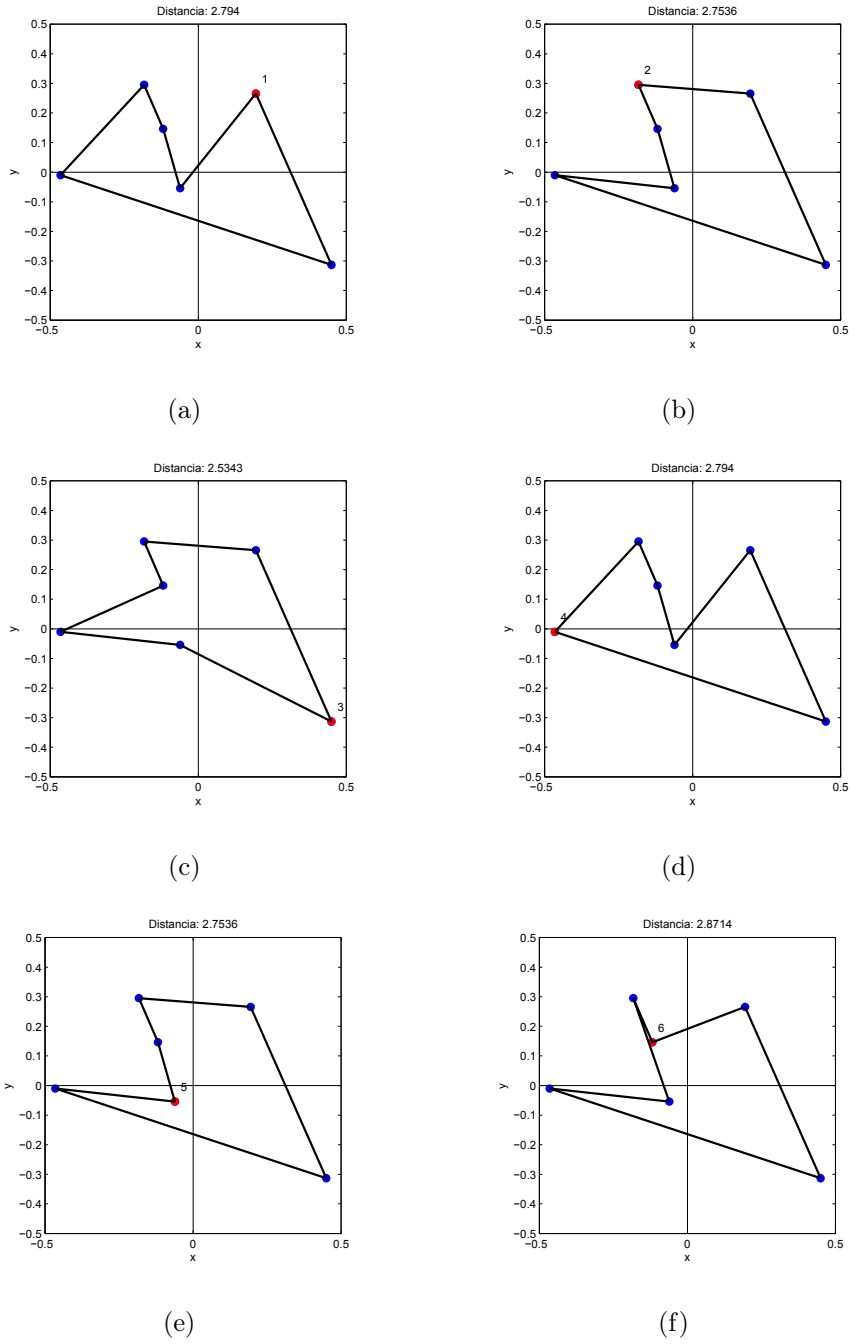


Figura 5.7: Posibles rutas óptimas a seguir en la red de 6 sensores distribuidos aleatoriamente de la Figura 5.1.

Por último, como ya hicimos en el apartado anterior, al implementar este método decidimos calcular la ruta óptima iniciando cada iteración con un sensor diferente, utilizando los N sensores de la red, de forma que finalmente obtuviéramos N rutas posibles,

y eligiendo la de menor longitud. En la Figura 5.7 se muestran las rutas obtenidas en dicha red junto con la distancia recorrida, observando que el inicio de la misma es diferente para cada caso (marcado en rojo) y que la menor distancia es de 2.5343, correspondiente a la Figura 5.7(c). Como puede verse esta ruta se aproxima mucho a la óptima vista en la Figura 5.2, aunque no coincide exactamente.

5.4.2. Método de inserción más cercano modificado.

En este apartado explicaremos un nuevo método de inserción, basado en el anterior, al que le hemos modificado una de sus características con el fin de reducir el coste computacional. En el método anterior, cuando teníamos más de un nodo en T, para buscar el nuevo nodo a insertar k, calculábamos la distancia desde todos los sensores pertenecientes a T a todos los sensores pertencientes al conjunto $\mathcal{N}\backslash T$. La modificación realizada en este método consiste simplemente en calcular sólo las distancias desde el último nodo de la ruta T hacia los sensores que pertenecen al conjunto $\mathcal{N}\backslash T$. El resto de pasos serían como hemos explicado anteriormente.

El objetivo de este nuevo método es, fundamentalmente, reducir el coste computacional, ya que no sería viable buscar la mínima distancia de todos los sensores añadidos en la lista hacia todos los sensores que quedan por añadir en una WSN densa con un gran número de nodos. De esta forma pasamos de $t \times (N-t)$ operaciones por iteración, donde t es el número de nodos en T y $N-t$ el número de nodos en el conjunto $\mathcal{N}\backslash T$, a sólo $N-t$ operaciones. Además, este cambio no afecta mucho a los resultados obtenidos, ya que las distancias finales son aproximadamente las mismas, como veremos en el Capítulo 6. Por tanto, en este método la ruta a seguir se construye como se muestra en el Algoritmo 5.2.

En la Figura 5.8 se observa un ejemplo de la generación de la ruta para la red de 6 sensores con la que estamos trabajando. El proceso en las Figuras 5.8(a) y 5.8(b) consiste en encontrar los 2 nodos más cercanos al primer sensor y establecer la ruta más corta para pasar por esos 3 sensores, comenzando siempre por el primer sensor seleccionado aleatoriamente, del mismo modo que hicimos en el método de inserción explicado anteriormente.

1. Elegir un nodo aleatoriamente.
2. Buscar el nodo más cercano al último nodo añadido a la ruta.
3. Insertar el nodo en la lista, calculando:

$$D_2 = \text{mín}\left\{d(a_i, k) + d(k, a_{i+1}) - d(a_i, a_{i+1})\right\}. \tag{5.17}$$

4. Conectar el primer y último sensor de la lista, una vez se han insertado todos los nodos de los que se dispone.

Algoritmo 5.3: Enumeración de los pasos a realizar en el algoritmo del método de inserción del vecino más cercano modificado.

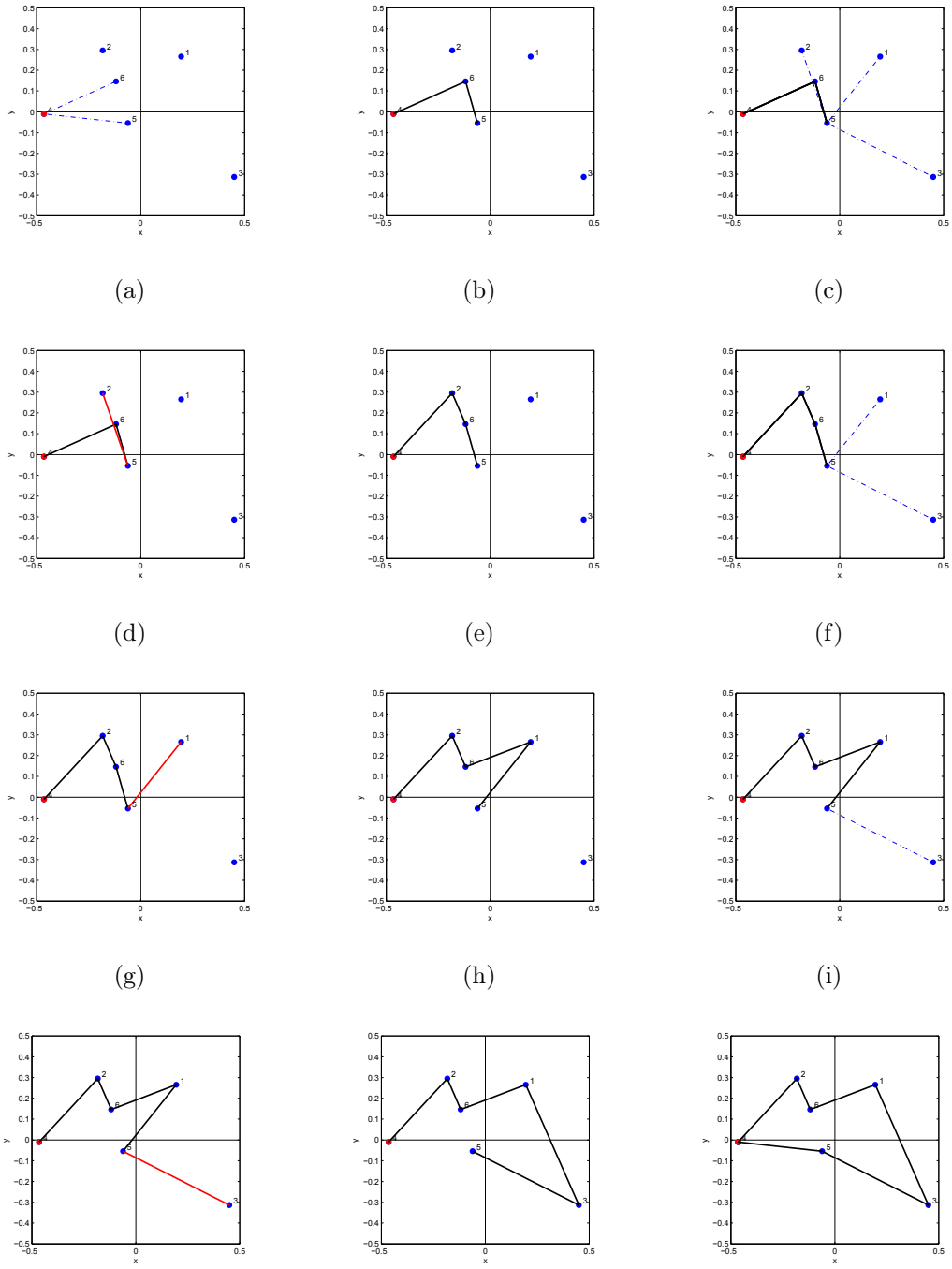
Las diferencias más importantes con respecto a dicho método comienzan a partir de la Figura 5.8(c), al añadir el tercer nodo, ya que sólo buscamos la distancia más corta desde el último nodo de la lista al resto de sensores aún disponibles, insertando dicho nodo en la lista teniendo en cuenta la Ecuación (5.6).

Una vez más, al implementar este método calcularemos la ruta óptima iniciando cada iteración con un sensor diferente y utilizando todos los nodos de la red, de forma que finalmente obtengamos N rutas posibles, seleccionando la de menor longitud. En la Figura 5.9 se muestran las rutas obtenidas en dicha red junto con la distancia recorrida, observando que la menor distancia es aproximadamente 2,5260 correspondiente a la Figura 5.9(d). Esta ruta coincide con la ruta óptima para esta red de sensores vista en la Figura 5.2.

5.5. Discusión

En este capítulo hemos estudiado tres alternativas para poder calcular la ruta óptima que debemos seguir para, partiendo de un nodo, regresar a él después de haber visitado todos los nodos de la red una sola vez, de forma que la distancia recorrida sea la menor posible.

En primer lugar, hemos descrito el funcionamiento del método del vecino más próximo,



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 5.8: Creación de la ruta óptima en la red de 6 sensores distribuidos aleatoriamente de la Figura 5.1 usando el método de inserción del vecino más cercano modificado.

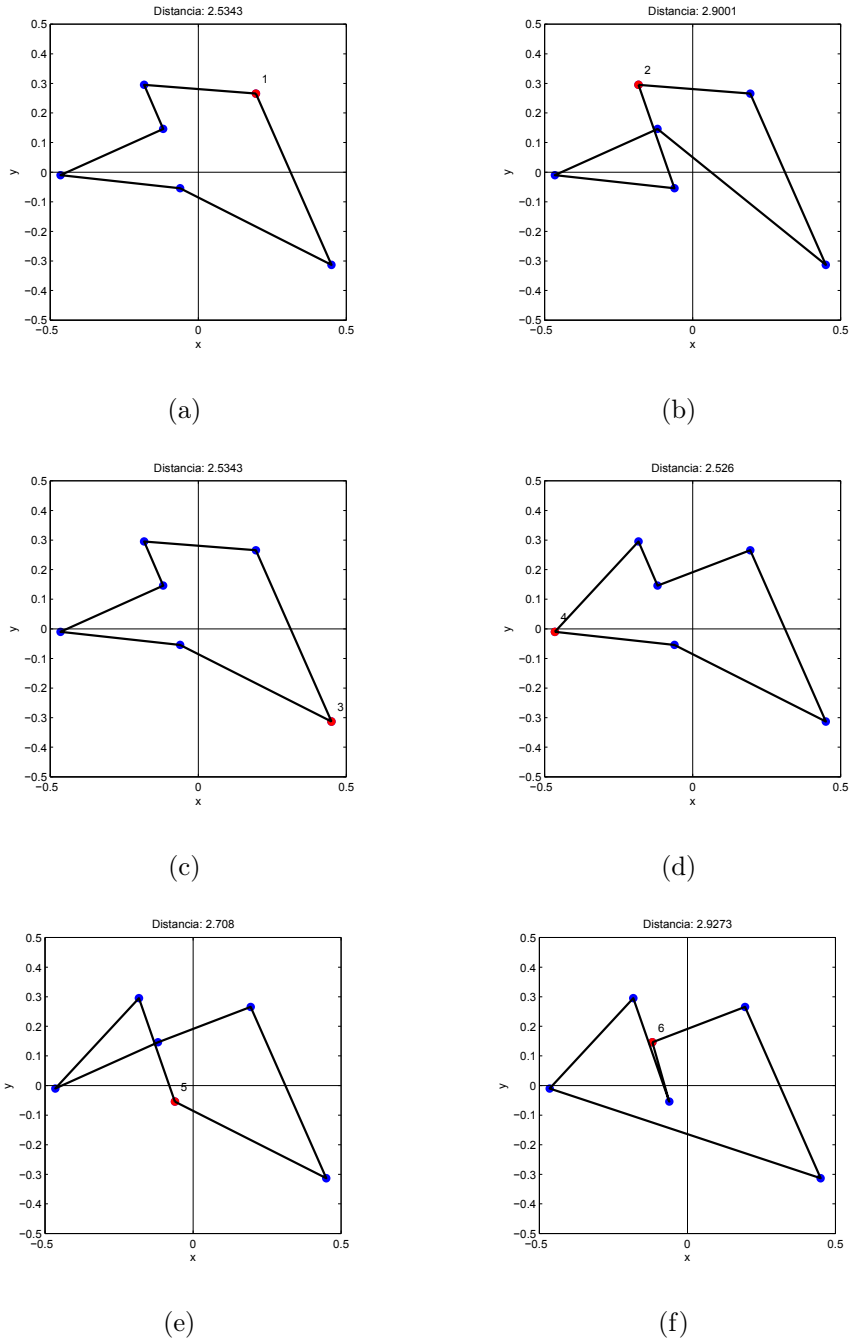


Figura 5.9: Posibles rutas óptimas a seguir en la red de 6 sensores distribuidos aleatoriamente de la Figura 5.1 usando el método de inserción más cercano modificado.

que consiste básicamente en buscar el nodo más cercano al último añadido. El coste computacional de este algoritmo no es mucho mayor que el del resto, e incluso puede llegar a ser menor que el de uno de ellos, como veremos en el capítulo siguiente, aunque uno

de los inconvenientes que presenta es que la distancia acumulada para la ruta encontrada puede ser muy superior a la de la ruta óptima para redes grandes.

En segundo lugar, hemos explicado el método de inserción, cuyo objetivo en cada paso es encontrar el nodo más cercano a cualquiera de los nodos añadidos a la ruta final e insertarlo en la misma. Como ya hemos comentado, la principal diferencia y ventaja de este método es que el nodo que queremos añadir no lo situaremos al final de la ruta, sino que podremos insertarlo en mitad de la lista. Esto hace que, en algunos casos, las distancias obtenidas sean menores y que las rutas contengan menos bucles.

El problema del método anterior es que su implementación resulta inviable en una red de sensores. En consecuencia, hemos añadido una modificación al método de inserción comentado anteriormente, que consiste en buscar el nodo más cercano sólo al último de la lista e insertarlo en la posición indicada por (5.6). De esta forma, el coste computacional va a disminuir considerablemente con respecto al método anterior, aunque las distancias obtenidas no difieren mucho de las obtenidas con el método de inserción inicial, como se verá en el Capítulo 6.

Capítulo 6

Simulaciones

6.1. Introducción

Una vez descritos los algoritmos de procesado distribuido que queremos implementar en este Proyecto Final de Carrera, vamos a exponer los resultados obtenidos en las simulaciones realizadas con cada uno de ellos. Los datos con los que vamos a trabajar los generaremos de forma sintética, ya que no disponemos de una red de sensores real.

El contenido de este capítulo está dividido en cuatro secciones:

- En la Sección 6.2 se describe el modo de llevar a cabo la generación de los datos sintéticos usados en las simulaciones.
- En la Sección 6.3 se muestran los resultados obtenidos en las diferentes pruebas realizadas para el problema del viajante.
- En la Sección 6.4 se exponen los resultados obtenidos para la estimación distribuida usando el algoritmo del subgradiente.
- En la Sección 6.5 se presentan los resultados obtenidos para la estimación distribuida usando el algoritmo de proyecciones alternas.

6.2. Generación de los datos sintéticos

En este apartado vamos a explicar los pasos seguidos para obtener los datos sintéticos con los que vamos a trabajar. Dado que no tenemos una red de sensores con la que realizar los experimentos, hemos tenido que simular las medidas tomadas por los mismos con ayuda de MATLAB. Considérese una red de N sensores distribuidos aleatoriamente en un área determinada ($X_1 \times X_2$), en la que suponemos que la magnitud a medir es la temperatura y donde el campo de temperatura presenta una forma del tipo Gaussiano, tal y como se indica en (6.1):

$$K(\mathbf{x}) = y_0 \ \exp\left(-\frac{1}{2}\ \left(\lambda_{11}\ x_1^2 + \lambda_{22}\ x_2^2 + 2\ \lambda_{12}\ x_1\ x_2\right)\right), \tag{6.1}$$

donde $y_0 \ = \ K\,[0,0]$ es la temperatura del foco, que, en nuestro caso, se encontrará en el centro del área; $\mathbf{x} = [x_1,\ x_2]^\top$ es la posición de medida; y λ_{11}, λ_{22} y λ_{12} son los parámetros que definen cómo se distribuye la temperatura en el área.

Para tener en cuenta las perturbaciones externas y los errores de medida, le añadimos un ruido aditivo blanco Gaussiano (AWGN, "Additive White Gaussian Noise") al campo de temperatura Gaussiano de la Ecuación (6.1), de modo que las observaciones realmente vendrán dadas por:

$$y(\mathbf{x}) = K(\mathbf{x})\ +\ \omega\,(\mathbf{x})\,, \tag{6.2}$$

donde $y(\mathbf{x})$ o simplemente y son las medidas de temperatura simuladas con ruido; $\omega\,(\mathbf{x})\ \sim N\,(\mathbf{0},\ \sigma^2\mathbf{I})$ representa el ruido (AWGN) de media nula y de varianza σ^2, e $\mathbf{I}$ denota la matriz identidad de dimensión $2\ \times\ 2$.

En (6.1) observamos que los datos simulados van a venir determinados por las posiciones de los nodos y los parámetros y_0, λ_{11}, λ_{22} y λ_{12}. En relación con estos, podemos diferenciar tres casos:

- $\lambda_{11} = \lambda_{22} = \lambda$ y $\lambda_{12} = 0$: en este caso (6.1) quedaría simplemente como

$$K(\mathbf{x}) = y_0 \exp\left(-\frac{1}{2}\lambda \left(x_1^2 + x_2^2\right)\right). \tag{6.3}$$

Como puede verse en la Figura 6.1, el valor de $K(\mathbf{x})$ depende por igual de la distancia con respecto al origen en ambos ejes. En consecuencia, las curvas de nivel van a ser círculos concéntricos alrededor del mismo.

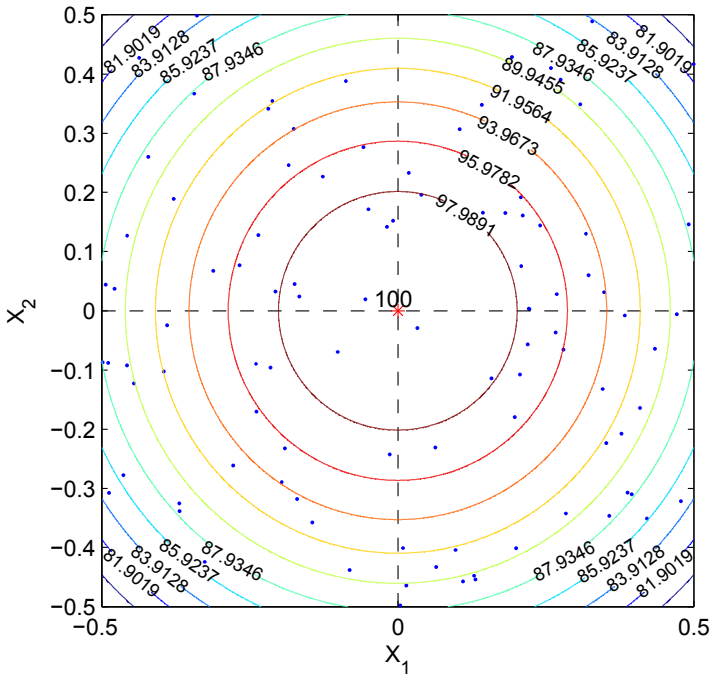


Figura 6.1: Distribución de la temperatura para $y_0 = 100$°C, $\lambda_{11} = \lambda_{22} = \lambda = 1$ y $\lambda_{12} = 0$, sin varianza de ruido añadida.

- $\lambda_{11} \neq \lambda_{22} \neq 0$ y $\lambda_{12} = 0$: ahora (6.1) quedaría como

$$K(\mathbf{x}) = y_0 \exp\left(-\frac{1}{2}\left(\lambda_{11}\, x_1^2 + \lambda_{22}\, x_2^2\right)\right). \tag{6.4}$$

En la Figura 6.2 se muestra un ejemplo de la distribución de la temperatura para este caso junto con los valores de λ_{11} y λ_{22} elegidos. En este caso, las curvas de

nivel son elipses con sus dos ejes principales alineados con los ejes x e y, donde el decremento de la temperatura es más rápido en la dirección del eje menor.

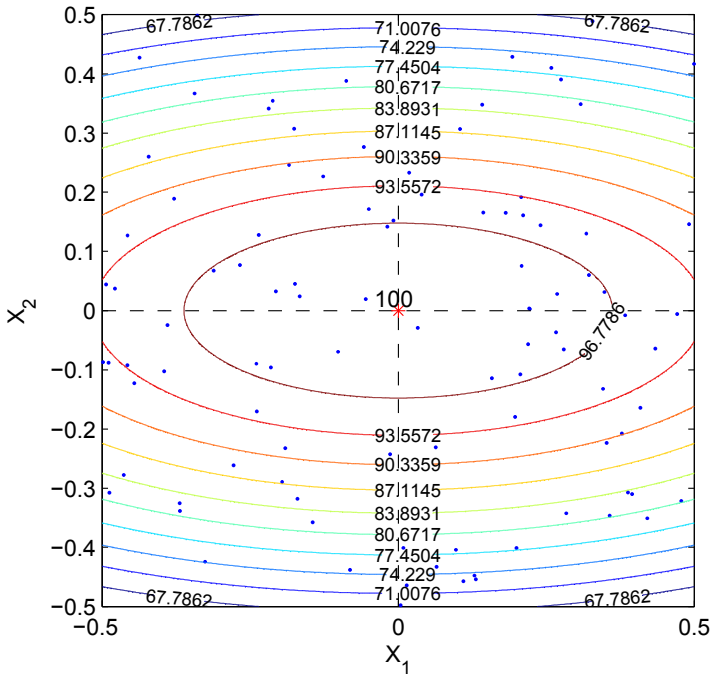


Figura 6.2: Distribución de la temperatura para $y_0 = 100°\text{C}$, $\lambda_{11} = 0{,}5$, $\lambda_{22} = 3$ y $\lambda_{12} = 0$, sin varianza de ruido añadida.

- $\lambda_{11} \neq \lambda_{22} \neq \lambda_{12} \neq 0$: en este caso el campo de temperatura presenta la forma dada por (6.1):

$$K(\mathbf{x}) = y_0 \ \exp\left(-\frac{1}{2}\ \left(\lambda_{11}\ x_1^2 + \lambda_{22}\ x_2^2 + 2\ \lambda_{12}\ x_1\ x_2\right)\right).$$

  Un ejemplo de la distribución de temperatura generada por estos valores se muestra en la Figura 6.3, en la que se observa que las curvas de nivel son elipses con sus dos ejes principales no alineados con los ejes x e y. De nuevo, el decremento de temperatura será más rápido en la dirección del eje menor.

A la hora de construir el campo de temperatura se proporcionan dos opciones para la especificación de los parámetros: directa e indirecta.

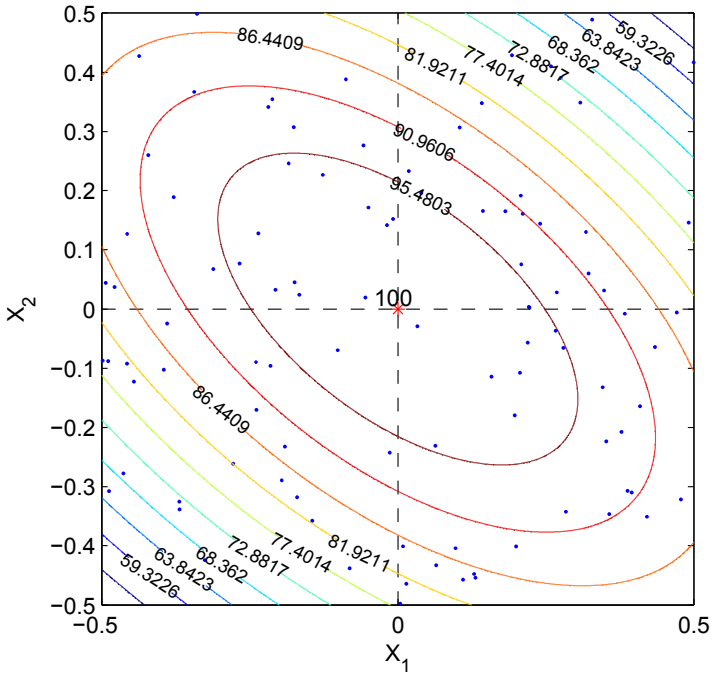


Figura 6.3: Distribución de la temperatura para $y_0 = 100°C$, $\lambda_{11} = 1{,}5$, $\lambda_{22} = 2{,}5$ y $\lambda_{12} = 1$ sin varianza de ruido añadida.

- **Directa:** Los parámetros de entrada a la función serán los valores y_0, λ_{11}, λ_{22} y λ_{12} proporcionados por el usuario.

- **Indirecta:** Los parámetros de entrada se fijarán a partir de la temperatura en el origen y del decremento de temperatura indicado por el usuario en cada eje $(\Delta T_1, \Delta T_2)$, así como del grado de inclinación (rotación) deseado, θ. Si el decremento de temperatura en ambos ejes fuera el mismo y sin inclinación, estaríamos ante la primera opción (esto es, $\lambda_{11} = \lambda_{22} = \lambda$ y $\lambda_{12} = 0$). En el caso de que fueran diferentes estaríamos ante la segunda situación (es decir, $\lambda_{11} \neq \lambda_{22}$ y $\lambda_{12} = 0$). Por último, si añadimos una inclinación nos encontraríamos ante la tercera opción ($\lambda_{11} \neq \lambda_{22} \neq \lambda_{12} \neq 0$). A partir de estos datos podremos obtener los valores de λ mencionados. Nótese que ΔT_1 hace referencia al decremento en el semieje positivo x, mientras que ΔT_2 hace referencia al decremento en el semieje positivo y. Esto es, si $X_1 = X_2 = [-0,5,\ 0,5]$, como en las Figuras 6.1-6.3, ΔT_1 será el decremento

en un $\Delta_x = 0{,}5$ y ΔT_2 será el decremento en un $\Delta_y = 0{,}5$.

A continuación, vamos a explicar cómo obtener el valor de los parámetros λ_{11}, λ_{22} y λ_{12} para cada caso, asumiendo que y_0 ya se ha fijado previamente a un valor normalizado igual a 1.

1. **Caso 1:** Los datos de entrada son $\Delta T_1 = \Delta T_2 = \Delta$ y $\theta = 0°$. Nótese que el valor fijado en esta opción es realmente ΔT_1, donde $\Delta T_1 = \Delta T_2 \Leftrightarrow L_{x_1} = L_{x_2}$. El parámetro λ se obtendrá a partir de 6.3, teniendo en cuenta que

$$K([0,0]) = y_0, \tag{6.5}$$

$$K([L_{x_1},0]) = y_0 \exp\left(-\frac{\lambda}{2}L_{x_1}^2\right). \tag{6.6}$$

Por tanto, el valor de ΔT, decremento de la temperatura en el semieje positivo x, será:

$$\Delta T = K([0,0]) - K([L_{x_1},0]) \quad = y_0\left(1 - \exp\left(-\frac{\lambda}{2}L_{x_1}^2\right)\right). \tag{6.7}$$

Reordenando términos en (6.7) y aplicando el logaritmo neperiano en ambos lados de la igualdad obtenemos:

$$\ln\left(\frac{y_0 - \Delta T}{y_0}\right) = -\frac{\lambda}{2}L_{x_1}^2. \tag{6.8}$$

Por último, despejando λ de (6.8) obtenemos que la expresión final será:

$$\lambda = -\frac{2}{L_{x1}^2}\ln\left(1 - \frac{\Delta T}{y_0}\right). \tag{6.9}$$

Nótese que en X_2 la temperatura sufrirá una caída de magnitud ΔT en una distancia L_{x_2}. Si $X_2 = [-L_{x_2}, L_{x_2}]$ con $L_{x_2} \neq L_{x_1}$, entonces $\Delta T_1 \neq \Delta T_2$. En la Figura 6.4 se muestra un ejemplo de la distribución de temperatura obtenida para este primer caso, donde el decremento de temperatura es el mismo en ambos ejes, ya que $L_{x_1} = L_{x_2}$. Además, podemos observar que, como explicamos anteriormente, las curvas de nivel son círculos concéntricos, verificándose que el decremento de temperatura es el deseado.

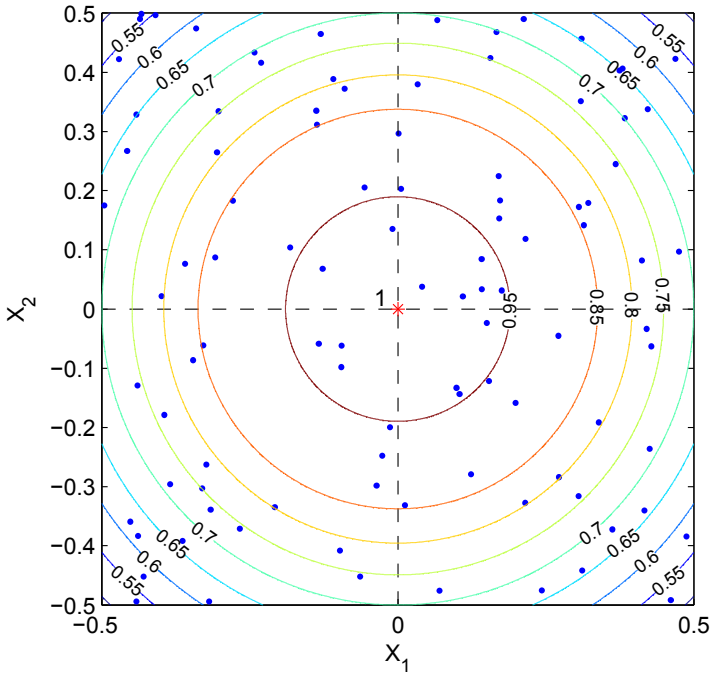


Figura 6.4: Distribución de temperatura normalizada obtenida para el caso 1, con $y_0 = 100°\text{C}$, $\Delta T_1 = \Delta T_2 = 30°C$ y $\theta = 0°$, sin varianza de ruido añadida.

2. **Caso 2:** Los datos de entrada son $\Delta T_1 \neq \Delta T_2 \neq 0°\text{C}$ y $\theta = 0°$. Los parámetros λ_{11} y λ_{22} se obtendrán teniendo en cuenta que $y_0 - \Delta T_1$, es la temperatura en los puntos $(L_{x_1}, 0)$ y $(-L_{x_1}, 0)$, e $y_0 - \Delta T_2$ es la temperatura en los puntos $(0, L_{x_2})$ y $(0, -L_{x_2})$. El desarrollo matemático será similar al del punto anterior. Las expresiones obtenidas para cada parámetro se muestran en las dos ecuaciones siguientes:

$$\lambda_{11} = -\frac{2}{L_{x_1}^2} \ln\left(1 - \frac{\Delta T_1}{y_0}\right), \tag{6.10}$$

$$\lambda_{22} = -\frac{2}{L_{x_2}^2} \ln\left(1 - \frac{\Delta T_2}{y_0}\right). \tag{6.11}$$

En este caso, la distribución quedaría como se muestra en la Figura 6.5, donde las curvas de nivel son elipses alineadas con los ejes x e y, tal y como indicamos ante-

riormente en el caso directo, y se comprueba como el decremento de la temperatura es el especificado en ambos ejes.

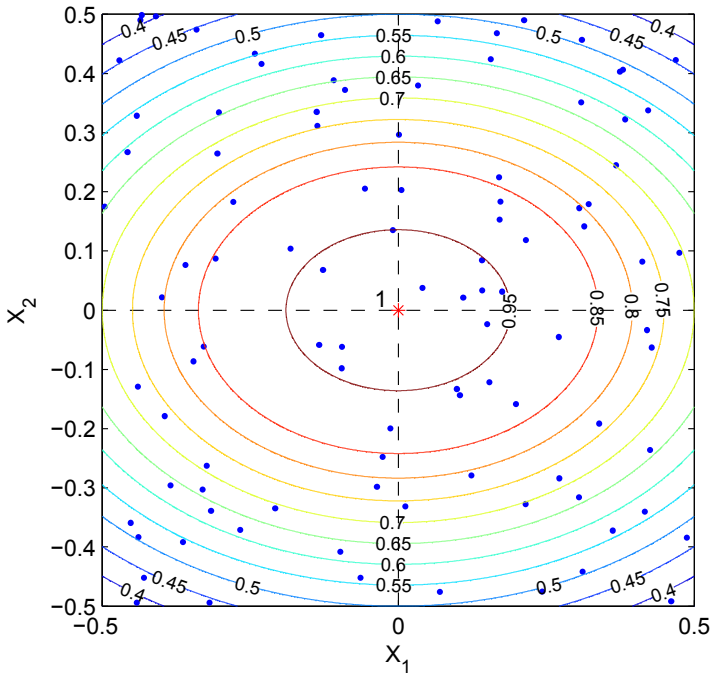


Figura 6.5: Distribución de temperatura normalizada obtenida para el caso 2, con $y_0 = 100°\text{C}$, $\Delta T_1 = 30°\text{C}$, $\Delta T_2 = 50°\text{C}$ y $\theta = 0°$, sin varianza de ruido añadida.

3. **Caso 3:** Los datos de entrada son $\Delta T_1 \neq \Delta T_2 \neq 0°\text{C}$ y $\theta \neq 0°$. En el caso anterior, en el punto $(L_{x1}, 0)$ el valor de la temperatura era $y_0 - \Delta T_1$. Ahora, al introducir un ángulo de inclinación θ queremos que esta temperatura esté en el punto $(x_1^{'}, x_2^{'})$. Igualmente, en el punto $(0, L_{x2})$ antes el valor de la temperatura era $y_0 - \Delta T_2$, mientras que ahora, debido a la inclinación mencionada, esta temperatura se alcanzará en el punto $(x_1^{''}, x_2^{''})$. En la Figura 6.6 se muestra un ejemplo de la rotación de los ejes. La transformación que debemos realizar para conseguir este objetivo se define como:

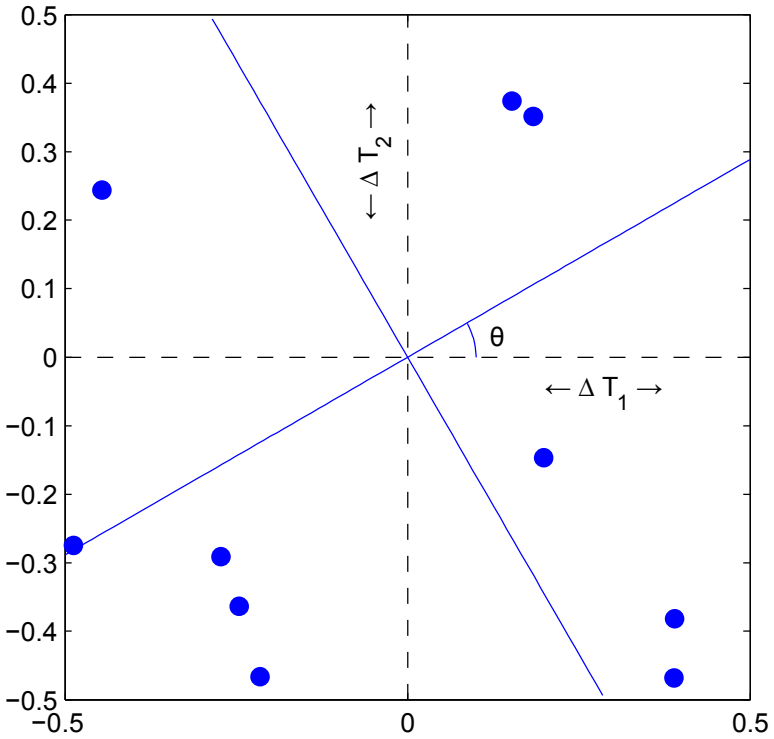


Figura 6.6: Rotación de los ejes para el caso 3 de la especificación indirecta de los parámetros.

$$\tilde{x}_1 = x_1 \cos\theta - x_2 \sin\theta, \tag{6.12}$$

$$\tilde{x}_2 = x_1 \sin\theta + x_2 \cos\theta. \tag{6.13}$$

Aplicando esta transformación obtenemos:

$$(L_{x_1}, 0) \;\rightarrow\; \left(x_1^{'}, x_2^{'}\right) \;=\; (L_{x_1}\cos\theta, L_{x_1}\sin\theta)\,. \tag{6.14}$$

$$(0, L_{x_2}) \;\rightarrow\; \left(x_1^{''}, x_2^{''}\right) \;=\; (-L_{x_2}\sin\theta, L_{x_2}\cos\theta)\,. \tag{6.15}$$

Para obtener los parámetros λ_{11}, λ_{22} y λ_{12}, vamos a suponer que conocemos la temperatura en los puntos $(L_{x1}, 0)$ y $(0, L_{x2})$, como hicimos en los casos anteriores, y, además, sabemos que, antes de la rotación, la temperatura en los puntos $(L_{x1},\ L_{x2})$

y $(L_{x1},\ -L_{x2})$ va a ser la misma. Por tanto, podríamos igualar ambas expresiones después de realizar la transformación correspondiente con estos nuevos puntos:

$$(L_{x_1}, L_{x_2}) \ \rightarrow \ (x_{11}^{'}, x_{21}^{'}) = (L_{x1}\cos\theta - L_{x2}\sin\theta,\ L_{x1}\sin\theta + L_{x2}\cos\theta). \tag{6.16}$$

$$(L_{x_1}, -L_{x_2}) \ \rightarrow \ (x_{12}^{'}, x_{22}^{'}) = (L_{x1}\cos\theta + L_{x2}\sin\theta,\ L_{x1}\sin\theta - L_{x2}\cos\theta). \tag{6.17}$$

Sustituyendo estos puntos en (6.1), e igualando las ecuaciones resultantes, obtenemos un valor de λ_{12} dependiente de λ_{11} y λ_{22}:

$$\lambda_{12} = \frac{(\lambda_{22} - \lambda_{11})\cos\theta\sin\theta}{\sin^2\theta - \cos^2\theta}. \tag{6.18}$$

Para obtener el valor de λ_{11} y λ_{22} volveremos a partir del valor de la temperatura en $(L_{x1}, 0)$ y $(0, L_{x2})$. De nuevo tendremos que realizar la transformación de estos dos puntos usando (6.14) y (6.15). Sustituyendo en (6.1) el valor de $(x_1^{'}, x_2^{'})$ y el parámetro λ_{12} obtenido en (6.18), obtenemos el siguiente valor para λ_{11}, dependiente a su vez de λ_{22}:

$$\lambda_{11} = \frac{(\lambda_{22}\sin^2\theta) - C_1(\sin^2\theta - \cos^2\theta)}{\cos^2\theta}. \tag{6.19}$$

A continuación, sustituiremos $(x_1^{''}, x_2^{''})$ y las expresiones de λ_{11} y λ_{12} que acabamos de desarrollar en (6.1), de forma que el parámetro λ_{22} se podrá definir como:

$$\lambda_{22} = C_1\sin^2\theta + C_2\cos^2\theta, \tag{6.20}$$

donde

$$C_1 = -\frac{2}{L_{x1}^2}\ln\left(1 - \frac{\Delta T_1}{y_0}\right), \tag{6.21}$$

$$C_2 = -\frac{2}{L_{x2}^2}\ln\left(1 - \frac{\Delta T_2}{y_0}\right). \tag{6.22}$$

Sustituyendo este valor en λ_{11}, llegamos a que su expresión final es:

$$\lambda_{11} = C_1 \cos^2\theta + C_2 \sin^2\theta. \tag{6.23}$$

Finalmente, una vez obtenidos estas dos expresiones, podremos obtener el parámetro λ_{12}:

$$\lambda_{12} = (C_1 - C_2)\cos\theta \sin\theta. \tag{6.24}$$

La distribución de temperatura para este tercer caso, con $\Delta T_1 = 30°\text{C}$, $\Delta T_2 = 50°\text{C}$ y $\theta = 45°$, sin ruido añadido a las medidas, quedaría como se muestra en la Figura 6.7. Una vez más, comprobamos que las curvas de nivel son elipses no alineadas con los ejes, tal y como ocurría en el caso directo, obteniéndose tanto la rotación como los descensos de temperatura previstos.

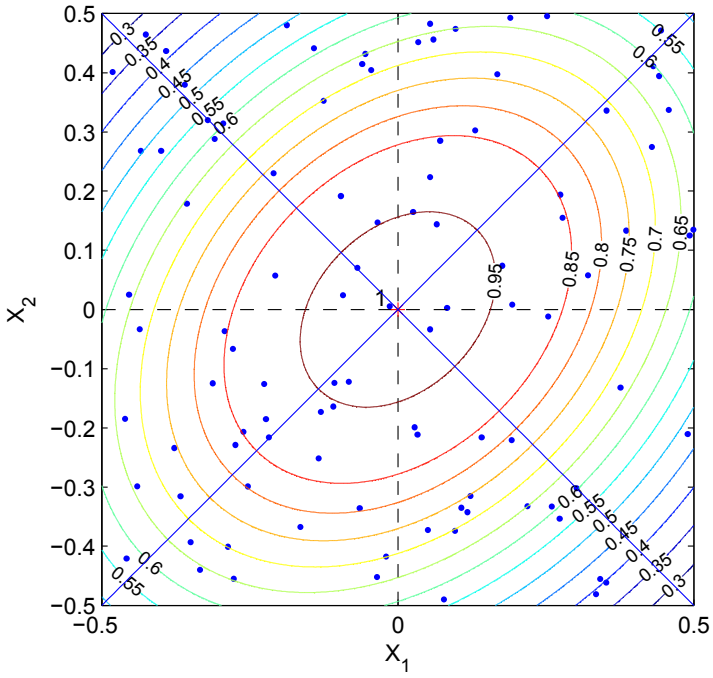


Figura 6.7: Distribución de temperatura normalizada obtenida para el caso 3 con $y_0 = 100°\text{C}$, $\Delta T_1 = 30°\text{C}$, $\Delta T_2 = 50°\text{C}$ y $\theta = 45°$, sin varianza de ruido añadida. En azúl tenemos marcados los ejes rotados $\theta = 45°$.

6.3. Problema del viajante: Vecino más próximo vs Métodos de inserción

En este apartado se pretende analizar el comportamiento del método de inserción modificado a la hora de generar la ruta a seguir por ambos algoritmos (subgradiente y proyecciones alternas). Mostraremos una serie de tablas con los resultados obtenidos en las diferentes simulaciones.

Para cada uno de los métodos descritos en el Capítulo 5 las simulaciones se llevaron a cabo generando N posiciones de sensores (entre 4 y 100) distribuidas uniformemente dentro de un área normalizada entre $-0,5$ y $0,5$ para un total de 10000 simulaciones. Debemos tener en cuenta que las distancias mostradas en todas las tablas tienen resultados normalizados. Esto es, las medidas de distancias se obtienen en base a las posiciones de los sensores, que a su vez son posiciones normalizadas entre $-0,5$ y $0,5$, tal y como acabamos de comentar.

En primer lugar se realizaron simulaciones con pocos sensores (entre 4 y 8), obteniéndose los resultados que podemos ver en la Tabla 6.1. En estos casos cabe destacar que, al tener tan pocos sensores, resulta viable encontrar la ruta óptima mediante búsqueda exhaustiva. Dicho algoritmo evalúa todos los posibles caminos y, como resultado final, nos devuelve aquel camino cuya longitud sea la menor.

N	N_{sim}	**AVP**	**AI**	**AIM**	**AO**
4	10000	1.9000 ± 0.2421	1.9000 ± 0.2421	1.8963 ± 0.2358	1.8940 ± 0.2307
5	10000	2.1402 ± 0.2427	2.1376 ± 0.2404	2.1280 ± 0.2308	2.1239 ± 0.2260
6	10000	2.3425 ± 0.2346	2.3363 ± 0.2310	2.3200 ± 0.2196	2.3138 ± 0.2154
7	10000	2.5127 ± 0.2222	2.5012 ± 0.2169	2.4823 ± 0.2058	2.4738 ± 0.2022
8	1000	2.5169 ± 0.2221	2.5053 ± 0.2167	2.4863 ± 0.2056	2.4776 ± 0.2020

Tabla 6.1: Valor medio y desviación típica de las distancias obtenidas con cada método: vecino más próximo (AVP), método de inserción (AI), método de inserción modificado (AIM) y algoritmo óptimo (AO). N indica el número de sensores de la red y N_{sim} el número de simulaciones realizadas.

En segundo lugar, nótese que el número de iteraciones se redujo a 1000 en redes de 8 sensores para que el tiempo de ejecución fuera razonable, ya que el número de caminos a evaluar aumenta exponencialmente con el número de sensores. Respecto a los resultados obtenidos, las columnas del método de inserción modificado y óptimo están marcadas en rojo por ser los métodos que proporcionan los mejores resultados, aunque la mejora con respecto al vecino más próximo o el método de inserción es pequeña. En cuanto a la desviación típica, podemos ver que permanece aproximadamente constante con independencia del número de sensores de la red y método seleccionado.

Al explicar los métodos del vecino próximo y de inserción, expusimos que la relación entre la distancia encontrada por cada método y la proporcionada por el método óptimo estaba acotada por (5.1) para el vecino próximo y por (5.3) para el método de inserción. En la Tabla 6.2 se pueden ver los resultados obtenidos al calcular la media (sobre las 10000 simulaciones) de la relación existente entre dichas distancias, junto con su desviación típica. Podemos comprobar que ambas medidas son aproximadamente constantes para un número fijo de sensores en la red, independientemente del método que estemos analizando. Sin embargo, sí que existe un pequeño incremento en el valor de las mismas a medida que el número de sensores de la red aumenta.

N	$\frac{d_{NN}}{d_{OPT}}$	$\frac{d_{INS}}{d_{OPT}}$	$\frac{d_{IM}}{d_{OPT}}$
4	1,0033 ± 0,0150	1,0033 ± 0,0150	1,0011 ± 0,0074
5	1,0080 ± 0,0229	1,0066 ± 0,0204	1,0017 ± 0,0089
6	1,0123 ± 0,0274	1,0096 ± 0,0243	1,0026 ± 0,0107
7	1,0160 ± 0,0302	1,0115 ± 0,0259	1.0034 ± 0,0117
8	1,0197 ± 0,0346	1,0114 ± 0,0246	1.0039 ± 0,0120

Tabla 6.2: Media y desviación típica de la relación entre las distancias del vécino próximo y los métodos de inserción vs el método óptimo.

Por otro lado, el valor de cota máximo para esta relación en función del número de nodos de la red se presenta en la Tabla 6.3, junto con el valor máximo alcanzado en las 10000 simulaciones. Al examinar los resultados obtenidos observamos que ninguno de ellos

excede el valor de la cota correspondiente, ya que el mínimo de la misma es 1,5 y el valor máximo alcanzado en las simulaciones no llega a 1,3. Comparando los valores máximos de la Tabla 6.3 con las cotas observamos que son muy inferiores a las mismas y a su vez muy superiores a los valores promedio de la Tabla 6.2. Además, se observa cómo el método de inserción modificado proporciona los mejores resultados, no sólo en valor máximo, sino también en promedio.

N	**cota** $\left\{\frac{d_{NN}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{NN}}{d_{OPT}}\right\}$	**cota** $\left\{\frac{d_{INS}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{INS}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{IM}}{d_{OPT}}\right\}$
4	1.5	1,1727	2	1,1727	1,1251
5	2	1,1846	2	1,1806	1,1193
6	2	1,2429	2	1,1881	1,1241
7	2	1,2211	2	1,2100	1,1545
8	2	1,2643	2	1,2323	1,1901

Tabla 6.3: Relación máxima permitida entre la media de las distancias de los distintos métodos implementados vs el óptimo, y cotas teóricas.

A continuación decidimos aumentar el tamaño de la red de sensores. En la Tabla 6.4 se muestran los resultados para redes con un número de sensores entre 20 y 100. Nótese que no se puede implementar el algoritmo óptimo para redes con este número de sensores, ya que resulta excesivamente costoso. A diferencia de las redes con un número de sensores pequeño, el método del vecino próximo nos proporciona en algunas ocasiones el camino óptimo, aunque el rendimiento de los métodos de inserción nunca es muy superior en general.

Hasta el momento hemos estado buscando la mejor ruta partiendo desde todos los sensores y eligiendo la de menor longitud. Sin embargo, en la Tabla 6.5 vamos a mostrar los resultados obtenidos al calcular las distancias recorridas por cada método al empezar por un nodo aleatoriamente en función del número de sensores en la red, para cada uno de los casos estudiados hemos marcado en rojo el valor mínimo de la distancia. En estos resultados observamos que, salvo para la red con $N = 7$, la ruta de menor coste nos la ofrece el método de inserción que hemos modificado en el desarrollo de este Proyecto

N	N_{sim}	AVP	AI	AIM
20	1000	4.2892 ± 0.1354	4.4178 ± 0.1374	4.4453 ±0.1378
30	1000	4.8269 ± 0.1151	5.3152 ± 0.1185	4.8993 ± 0.1160
50	1000	5.9457 ± 0.0926	6.2304 ± 0.0789	5.4841 ± 0.0626
70	1000	6.6620 ± 0.0651	7.0019 ± 0.0607	7.2913 ± 0.0710
90	1000	9.0020 ± 0.0824	8.5760 ± 0.0485	8.1450 ± 0.0515
100	1000	9.464 ± 0.0765	8.5516 ± 0.0536	8.6456 ± 0.0639

Tabla 6.4: Valor medio de las distancias para cada método junto con su desviación típica.

Final de Carrera a partir del método de inserción y que explicamos detalladamente en el Capítulo 5.

N	Número de iteraciones	AVP	AI	AIM
4	1000	1.9205 ± 0.2469	1.9267 ± 0.2461	1.9142 ± 0.2428
5	1000	2.2109 ± 0.2577	2.2076 ± 0.2569	2.1949 ± 0.2473
6	1000	2.4519 ± 0.2541	2.4425 ± 0.2548	2.4372 ± 0.2424
7	1000	2.6406 ± 0.2466	2.6213 ± 0.2451	2.6266 ± 0.2331
8	1000	2.8590 ± 0.2432	2.8183 ± 0.2381	2.8168 ± 0.2253
20	1000	4.4997 ± 0.1908	4.3803 ± 0.1716	4.3115 ± 0.1572
30	1000	5.5100 ± 0.1692	5.3841 ± 0.1432	5.2532 ± 0.1322
50	1000	6.9910 ± 0.1429	6.8346 ± 0.0117	6.7110 ± 0.1073
70	1000	8.2307 ± 0.1268	8.0359 ± 0.0928	7.9260 ± 0.0905
90	1000	9.2331 ± 0.1149	9.0429 ± 0.0789	8.9938 ± 0.0817
100	1000	9.6922 ± 0.1097	9.5156 ± 0.0745	9.4472 ± 0.0763

Tabla 6.5: Valor medio de las distancias para cada método junto con su desviación típica para el caso en que empecemos la ruta por un nodo escogido aleatoriamente.

A continuación, vamos a calcular, una vez más, la relación existente entre la distancia de cada método estudiado y la distancia del método óptimo (al igual que antes, sólo para redes con un número de sensores entre 4 y 8). En la Tabla 6.6 se pueden ver la media y

la desviación típica para cada caso, donde ambas medidas tienen ligeras diferencias según el número de sensores de la red. Una vez más, en la Tabla 6.7 comprobamos que la cota máxima asociada a cada método no es excedida por ninguno de ellos, cumpliéndose de nuevo nuestras expectativas. Además, nuevamente el método de inserción modificado es el que proporciona los mejores valores tanto de distancia media como máxima.

N	$\frac{d_{NN}}{d_{OPT}}$	$\frac{d_{INS}}{d_{OPT}}$	$\frac{d_{IM}}{d_{OPT}}$
4	1,0198 ± 0,0369	1.0230 ± 0,0442	1,0164 ± 0,0337
5	1,0405 ± 0,0535	1.0412 ± 0,0552	1,0354 ± 0,0500
6	1,0580 ± 0,0646	1.0564 ± 0,0626	1,0519 ± 0,0602
7	1,0724 ± 0.0740	1,0640 ± 0,0660	1,0627 ± 0,0658
8	1,0847 ± 0,0799	1,0750 ± 0,0709	1,0743 ± 0,0695

Tabla 6.6: Media y desviación típica de la relación entre las distancias del vécino próximo y los métodos de inserción vs el método óptimo para el caso en que evaluamos la ruta empezando por un nodo elegido arbitraroamente.

N	**cota** $\left\{\frac{d_{NN}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{NN}}{d_{OPT}}\right\}$	**cota** $\left\{\frac{d_{INS}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{INS}}{d_{OPT}}\right\}$	máx $\left\{\frac{d_{IM}}{d_{OPT}}\right\}$
4	1.5	1,2249	2	1,2283	1,2249
5	2	1,3428	2	1,3653	1,3138
6	2	1,4054	2	1,3780	1,3369
7	2	1,5465	2	1,3976	1,3741
8	2	1,4788	2	1,3326	1,3483

Tabla 6.7: Relación máxima permitida entre la media de las distancias de los distintos métodos implementados vs el óptimo, y cotas teóricas para el caso en que evaluamos la ruta empezando por un nodo escogido aleatoriamente.

Al realizar las simulaciones pudimos comprobar que en cualquiera de los métodos que estamos estudiando las distancias de salto entre cada nodo disminuyen a medida que aumentamos el número de sensores de la red, tal y como era de esperar. A modo de ejemplo, en la Figura 6.8 se muestra dicho comportamiento en el método de inserción más

cercano.

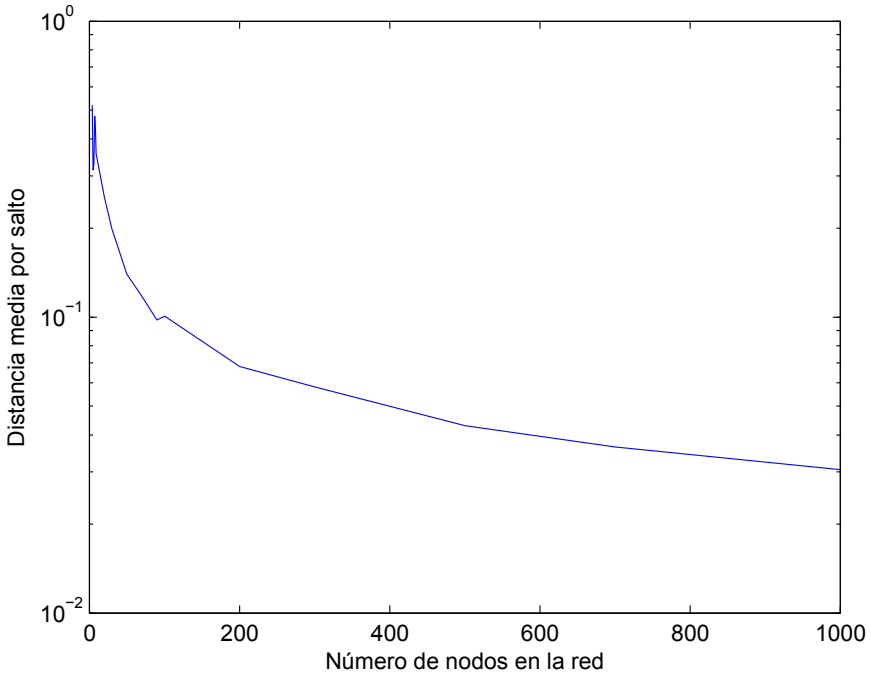


Figura 6.8: Evolución de la distancia media por salto con el aumento de sensores en la red para el método de inserción más cercano.

En las Figuras 6.9(a), 6.9(b) y 6.9(c) se muestra la ruta óptima obtenida con cada método empezando por todos los nodos posibles: vecino próximo (Figura 6.9(a)), inserción (Figura 6.9(b)) e inserción modificado (Figura 6.9(c)). De ellas concluimos que el método de inserción modificado proporciona la menor longitud. En las Figuras 6.9(d), 6.9(e) y 6.9(f) se observa la ruta óptima conseguida con cada método empezando por un nodo aleatorio: vecino próximo (Figura 6.9(d)), inserción (Figura 6.9(e)) e inserción modificado (Figura 6.9(f)). En este caso, el camino de menor longitud viene dado por el método de inserción.

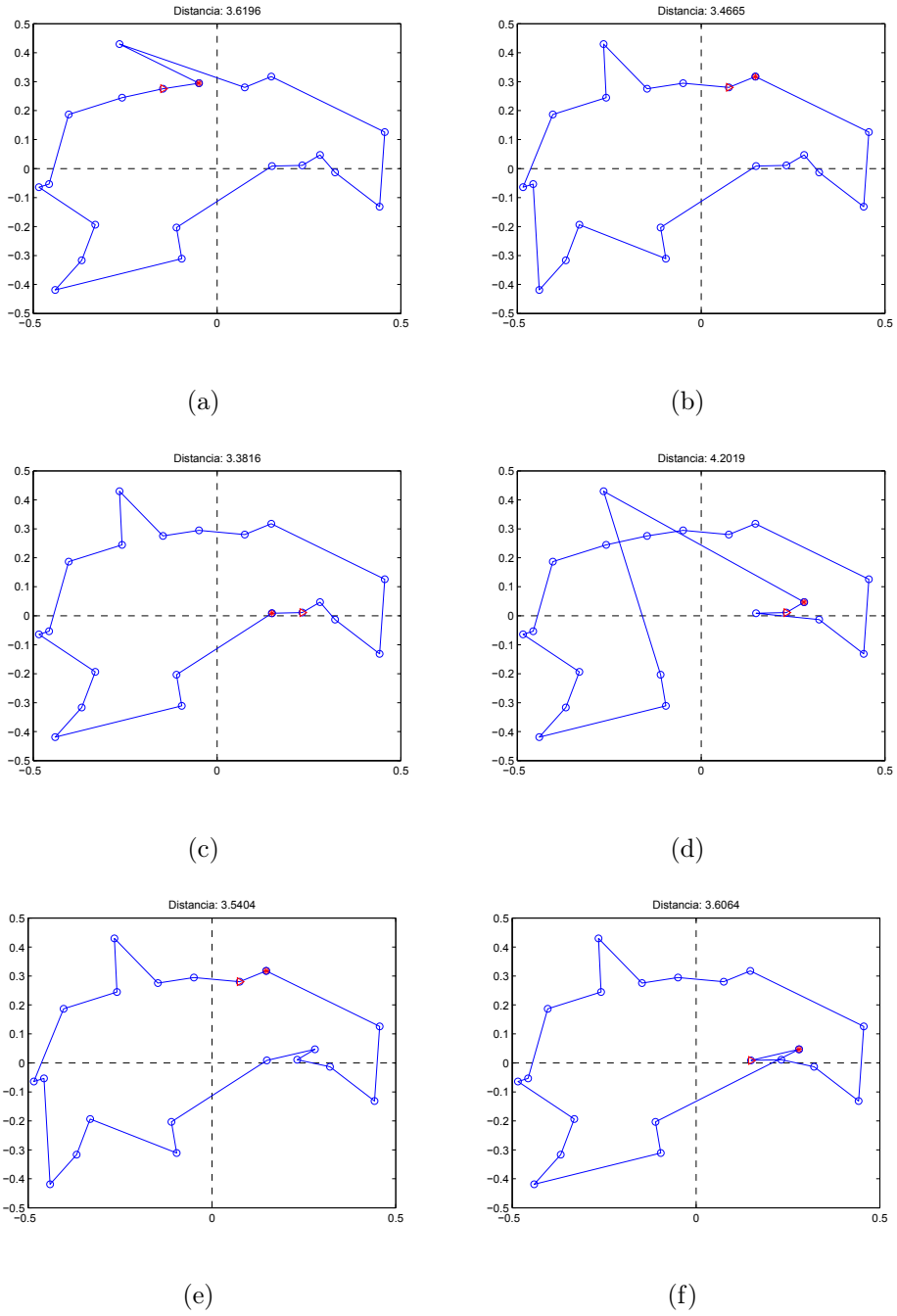


Figura 6.9: Ruta elegida por cada uno de los métodos para una red de 20 sensores distribuidos aleatoriamente. El primer y segundo sensor de la lista están marcados en rojo con triángulo y un asterisco, respectivamente. (a) vecino próximo (AVP) empezando por todos los posibles nodos, (b) inserción más cercano (AI) empezando por todos los posibles nodos, (c) inserción más cercano modificado (AIM) empezando por todos los posibles nodos, (d) AVP empezando por un nodo aleatorio, (e) AI empezando por un nodo aleatorio y (f) AIM empezando por un nodo aleatorio.

6.4. Optimización distribuida: Algoritmo subgradiente

En esta sección vamos a estudiar el primero de los algoritmos de estimación distribuida analizados, el algoritmo de subgradiente, que a su vez compararemos con el método del gradiente estudiado en el Capítulo 4. En la implementación del mismo necesitamos la ayuda de un algoritmo iterativo de optimización, de forma que podamos estimar los parámetros con los que poder calcular las medidas de temperatura en cualquier punto del área considerada. En nuestras simulaciones utilizaremos el método más sencillo y el más complejo de los estudiados en el Capítulo 4. El primero hace referencia al descenso de gradiente, para el cual hemos tenido que implementar específicamente una función, debido a que la función *graddesc* ofrecida por la librería Netlab de MATLAB proporcionaba una tasa de convergencia muy lenta. El segundo se refiere al gradiente conjugado escalado, cuyas simulaciones se han llevado a cabo utilizando la función *scg* de la librería Netlab de MATLAB [62] [1].

Las simulaciones se han realizado para redes con un número de sensores variable entre 100 y 1000, una varianza de ruido añadida a las medidas sintéticas de 0, 10^{-2} y 10^{-3}, y para 10000 valores distintos de los parámetros iniciales con distribución uniforme entre 0 y 10. En la Tabla 6.8 se muestran los parámetros reales a partir de los que hemos obtenido las medidas sintéticas de cada sensor. Nótese que las variables elegidas realmente han sido ΔT_1, ΔT_2, θ y la temperatura del foco (y_0), obteniéndose de forma indirecta los parámetros λ_{11}, λ_{22} y λ_{12}.

$\Delta\ T_1$(°C)	$\Delta\ T_2$(°C)	θ°	**Temperatura del foco** (°C)	λ_{11}	λ_{22}	λ_{12}
10	50	20	100	1,3929	4,9951	-1,5113

Tabla 6.8: Datos reales para las simulaciones del algoritmo de subgradiente.

Para analizar los parámetros obtenidos en las diferentes simulaciones hemos decidido calcular el sesgo, la varianza y el error cuadrático medio (MSE, "Mean Square Error").

[1] Las funciones de Netlab utilizadas en este capítulo se pueden descargar gratuitamente de http://www.mathworks.com/matlabcentral/fileexchange/2654-netlab

- En primer lugar, el sesgo nos indica la diferencia entre la media de los parámetros estimados, $\hat{\theta}(k)$, y el valor real de los mismos, θ, de forma que si el valor de dicha operación es nulo podremos decir que estamos ante un estimador insesgado, propiedad típicamente deseada en los estimadores, ya que implica haber conseguido una buena aproximación del valor real de los parámetros en promedio. Esta medida se puede expresar matemáticamente como

$$b\left(\hat{\theta}\right) = \frac{1}{M}\sum_{k=1}^{M}\hat{\theta}\left(k\right) - \theta, \tag{6.25}$$

  siendo M el número de estimas (simulaciones) realizadas.

- En segundo lugar, la varianza es una medida de dispersión que nos proporciona información acerca de la desviación con respecto a la media obtenida al estimar los parámetros. Matemáticamente se puede definir como

$$Var\left(\hat{\theta}\right) = \frac{1}{M}\sum_{k=1}^{M}\left(\hat{\theta}\left(k\right) - \theta\right)^{2}. \tag{6.26}$$

- En tercer y último lugar, el MSE nos proporciona el error cuadrático medio cometido al estimar los distintos parámetros. Se puede expresar matemáticamente como

$$MSE\left(\hat{\theta}\right) = b\left(\hat{\theta}\right)^{2} + Var\left(\hat{\theta}\right). \tag{6.27}$$

6.4.1. Algoritmo de gradiente

Antes de realizar el estudio del algoritmo de subgradiente analizaremos el método de gradiente descrito en el Capítulo 4 con el fin de compararlos. Para empezar, expondremos las tablas asociadas al gradiente conjugado escalado (Tablas 6.9, 6.10 y 6.11), donde se pueden analizar las medidas de sesgo, varianza y MSE, considerando σ^2_{ruido} la varianza de ruido añadida a los datos sintéticos.

$\sigma^2_{ruido} = 0$					
θ	N	100	200	500	1000
y_0	Sesgo	9,86e-6	3,33e-5	4,56e-5	1,86e-5
	Varianza	1,34e-7	1,26e-6	5,17e-6	3,56e-7
	MSE	1,34e-7	1,26e-6	5,18e-6	3,56e-7
λ_{11}	Sesgo	1,39e-4	6,73e-4	3,56e-4	3,12e-4
	Varianza	3,61e-5	1,76e-4	3,54e-4	1,28e-4
	MSE	3,61e-5	1,77e-4	3,54e-4	1,28e-4
λ_{22}	Sesgo	8,30e-5	3,80e-4	8,14e-4	9,85e-4
	Varianza	8,46e-5	8,86e-5	2,90e-4	1,30e-3
	MSE	8,46e-5	8,87e-5	2,91e-4	1,30e-3
λ_{12}	Sesgo	-4,24e-5	-3,94e-4	-1,30e-4	1,34e-4
	Varianza	1,39e-5	4,40e-4	4,86e-4	2,60e-3
	MSE	1,39e-5	4,40e-4	4,86e-4	2,60e-3

Tabla 6.9: Sesgo, varianza y MSE obtenidos al estimar los parámetros para diferentes tipos de redes con el método del gradiente conjugado escalado y $\sigma^2_{ruido} = 0$.

$\sigma^2_{ruido} = 10^{-3}$					
θ	N	100	200	500	1000
y_0	Sesgo	2,00e-4	2,66e-4	3,49e-5	9,00e-5
	Varianza	1,33e-7	2,33e-7	9,14e-8	1,16e-6
	MSE	1,74e-7	3,04e-7	9,26e-8	1,17e-6
λ_{11}	Sesgo	-1,30e-3	-2,90e-4	-1,40e-3	-1,40e-3
	Varianza	3,85e-5	3,93e-4	2,51e-5	2,17e-4
	MSE	4,01e-5	3,93e-4	2,71e-5	2,18e-4
λ_{22}	Sesgo	6,10e-3	3,90e-3	1,60e-3	3,90e-3
	Varianza	8,03e-5	5,81e-5	4,72e-5	1,50e-3
	MSE	1,17e-4	7,34e-5	4,99e-5	1,50e-3
λ_{12}	Sesgo	2,60e-3	-1,80e-3	2,97e-4	6,77e-4
	Varianza	1,45e-5	1,18e-4	1,79e-5	2,80e-3
	MSE	2,11e-5	1,21e-4	1,80e-5	2,80e-3

Tabla 6.10: Sesgo, varianza y MSE obtenidos al estimar los parámetros para diferentes tipos de redes con el método del gradiente conjugado escalado y $\sigma^2_{ruido} = 10^{-3}$.

$\sigma^2_{ruido} = 10^{-2}$					
θ	N	100	200	500	1000
y_0	Sesgo	-1,30e-3	-9,29e-4	1,80e-3	-8,18e-3
	Varianza	6,03e-7	1,01e-6	2,34e-6	8,76e-8
	MSE	2,28e-6	1,96e-6	5,71e-6	7,57e-7
λ_{11}	Sesgo	1,87e-2	-2,08e-2	3,88e-2	1,86e-2
	Varianza	7,41e-5	5,07e-4	7,15e-4	5,56e-5
	MSE	4,25e-4	9,37e-4	2,20e-3	4,03e-4
λ_{22}	Sesgo	-7,58e-2	1,70e-3	7,70e-3	-7,30e-3
	Varianza	1,57e-4	9,44e-4	4,28e-5	8,13e-5
	MSE	5,90e-3	9,47e-4	1,02e-4	1,35e-4
λ_{12}	Sesgo	-7,80e-3	7,04e-4	-1,28e-2	2,90e-3
	Varianza	1,47e-5	1,60e-3	7,59e-4	3,08e-4
	MSE	7,58e-5	1,60e-3	4,28e-3	8,00e-4

Tabla 6.11: Sesgo, varianza y MSE obtenidos al estimar los parámetros para diferentes tipos de redes con el método del gradiente conjugado escalado y $\sigma^2_{ruido} = 10^{-2}$.

A partir de estas tablas podemos extraer las siguientes conclusiones:

- Al analizar el sesgo mostrado en cada una de las tablas anteriores, observamos que dicho parámetro permanece aproximadamente constante para cada parámetro con independencia del número de sensores de la red para un mismo valor de σ^2_{ruido}. Además, podemos decir que con independencia del valor de σ^2_{ruido} el parámetro y_0 tiene valores de sesgo que están uno o dos órdenes de magnitud por debajo del sesgo del resto de parámetros.

- La varianza y el MSE se pueden analizar de forma conjunta, debido a que el valor del sesgo es muy bajo, lo que implica que ambas medidas coincidan. Independientemente del σ^2_{ruido} observamos que el parámetro y_0 ofrece valores para ambas medidas aproximadamente dos o tres órdenes de magnitud por debajo de la varianza y del MSE del resto de parámetros. Analizando λ_{11}, λ_{22} y λ_{12} por separado se observa

que los valores de estas medidas para dichos parámetros se encuentran entre 10^{-5} y 10^{-3}.

Entre los objetivos secundarios de este Proyecto Final de Carrera se encuentra el análisis de los métodos centralizados de optimización y su comparación con los métodos distribuidos. Una manera de cotejar estos algoritmos sería comparar la estimación final de los parámetros, tal y como se muestra en la Tabla 6.12, donde se observa que la estimación llevada a cabo por cada uno de los dos métodos analizados nos proporciona estimas muy cercanas a los valores reales de los parámetros estimados. Por tanto, deducimos que, desde el punto de vista de la estimación estos algoritmos son buenos, aunque no debemos olvidar que estamos ante métodos centralizados en los que el trabajo principal de la red es llevado a cabo por un nodo central, lo que implica una serie de inconvenientes vistos en el Capítulo 3. Los resultados para $\sigma^2_{ruido} = 10^{-2}$ y $\sigma^2_{ruido} = 10^{-3}$ (no mostrados) son muy similares.

	Descenso de gradiente				**Gradiente conjugado escalado**			
N	y_0	λ_{11}	λ_{22}	λ_{12}	y_0	λ_{11}	λ_{22}	λ_{12}
100	1,00	1.39	4,99	-1,51	0,99	1,37	4,90	-1,50
200	1,00	1,39	4,99	-1,51	1,00	1,39	4,91	-1,50
500	1,00	1,39	4,99	-1,51	1,00	1,39	4,92	-1,50
1000	1,00	1,39	4,99	-1,51	1,00	1,38	4,92	-1,50

Tabla 6.12: Valor de los parámetros estimados para el descenso de gradiente y el gradiente conjugado escalado con $\sigma^2_{ruido} = 0$.

6.4.2. Algoritmo de subgradiente

En este apartado se mostrarán los resultados obtenidos en las simulaciones realizadas con el algoritmo del subgradiente, utilizando el método del descenso de gradiente y el gradiente conjugado escalado para obtener las estimas locales. Antes de empezar este estudio debemos recordar la función de coste a minimizar que viene dada por (4.23) con un término de regularización nulo:

$$\arg \min_{\theta} \left\{ \frac{1}{N} \sum_{n=1}^{N} \left(f\left(\mathbf{x}_n; \theta\right) - y_n \right)^2 \right\}. \tag{6.28}$$

En las Tablas 6.13, 6.14 y 6.15 se exponen los resultados del sesgo, la varianza y el error cuadrático medio (MSE, "Mean square error") conseguidos al estimar los parámetros (y_0, λ_{11}, λ_{22} y λ_{12}) de la Tabla 6.8 con el método del gradiente conjugado escalado para diferentes varianzas de ruido. A continuación, comentaremos dichos resultados, tal y como hicimos en el apartado anterior.

- Al analizar el sesgo, nuevamente observamos que toma valores muy bajos (aproximadamente entre 10^{-3} y 10^{-2}). Los valores aquí obtenidos son mayores que con el método del gradiente, aunque podemos seguir considerando que el valor obtenido para el sesgo es despreciable y que el método del subgradiente proporciona medidas insesgadas.

- Al igual que antes, la varianza y el MSE se pueden estudiar de manera conjunta debido a que el sesgo toma valores muy bajos. Una vez más, el parámetro y_0 proporciona unos valores para estas medidas de aproximadamente dos o tres órdenes de magnitud por debajo de la varianza y el MSE del resto de parámetros.

Uno de los propósitos de este apartado es analizar el rendimiento del algoritmo del subgradiente en función del método de estimación centralizada utilizado localmente. En este sentido, existen varias medidas que nos podrían aportar información relevante sobre la calidad del estimador, tales como el valor de los parámetros estimados, el número de iteraciones y el tiempo de ejecución que necesita el algoritmo para converger. En la Tabla 6.16 se exponen los valores estimados de los parámetros para cada caso. Los resultado obtenidos para valores de σ^2_{ruido} igual a 10^{-2} o 10^{-3} son similiares a los de esta tabla, de modo que no se muestran. Comparando los datos de dicha tabla con los mostrados en la Tabla 6.8, consideramos que ambos algoritmos son buenos estimadores y que nos podrían proporcionar unas medidas de temperaturas muy próximas al valor real.

$\sigma_r^2 uido = 0$					
$\boldsymbol{\theta}$	N	100	200	500	1000
y_0	Sesgo	1,30e-3	-2,30e-3	-3,33e-4	7,30e-3
	Varianza	9,12e-5	1,72e-4	5,16e-5	1,30e-3
	MSE	9,30e-5	1,78e-4	5,16e-5	1,40e-3
λ_{11}	Sesgo	3,47e-2	-3,54e-2	-9,30e-3	7,24e-2
	Varianza	6,87e-2	0,13	3,60e-3	9,66e-2
	MSE	6,99e-2	0,13	3,70e-3	0,10
λ_{22}	Sesgo	1,83e-2	1,98e-2	-1,25e-2	-5,94e-2
	Varianza	3,09e-2	2,76e-2	1,64e-2	4,02e-2
	MSE	3,12e-2	2,80e-2	1,66e-2	4,38e-2
λ_{12}	Sesgo	3,72e-2	-7,90e-3	9,90e-3	2,60e-3
	Varianza	1,99e-2	9,84e-2	3,10e-3	2,90e-3
	MSE	1,99e-2	9,98e-2	3,20e-3	2,90e-3

Tabla 6.13: Sesgo, varianza y MSE obtenidos al estimar los parámetros para el subgrad. utilizando en la estimación centralizada el grad. conjugado escalado y $\sigma^2_{ruido} = 0$.

$\sigma^2_{ruido} = 10^{-3}$					
$\boldsymbol{\theta}$	N	100	200	500	1000
y_0	Sesgo	1,30e-3	-2,16e-4	6,71e-4	5,70e-3
	Varianza	4,85e-4	1,49e-5	8,77e-5	2,70e-3
	MSE	4,87e-4	1,49e-5	9,01e-5	2,70e-3
λ_{11}	Sesgo	3,70e-3	-4,11e-2	2,01e-2	7,09e-2
	Varianza	1,53e-2	5,67e-2	4,78e-2	0,14
	MSE	1,53e-2	5.84e-2	4,79e-2	0,15
λ_{22}	Sesgo	-1,60e-3	-3,20e-3	-1,98e-2	-8,28e-2
	Varianza	7,8e-3	1,68e-2	7,92e-2	5,22e-2
	MSE	7,80e-3	1,68e-2	7,90e-2	5,91e-2
λ_{12}	Sesgo	5,40e-3	2,61e-2	2,70e-3	1,22e-2
	Varianza	3,35e-4	2,40e-3	6,32e-3	4,50e-3
	MSE	3,65e-4	3,00e-3	6,29e-3	4,70e-3

Tabla 6.14: Sesgo, varianza y MSE obtenidos al estimar los parámetros para el subgrad. utilizando en la estimación centralizada el grad. conjugado escalado y $\sigma^2_{ruido} = 10^{-3}$.

$\sigma^2_{ruido} = 10^{-2}$					
$\boldsymbol{\theta}$	N	100	200	500	1000
y_0	Sesgo	5,90e-3	-3,50e-3	-6,81e-3	-2,80e-3
	Varianza	1,48e-4	2,08e-4	2,12e-5	1,30e-3
	MSE	1,83e-4	2,21e-4	2,13e-5	1,30e-3
λ_{11}	Sesgo	-7,03e-2	-8,36e-2	-9,21e-2	2,71e-2
	Varianza	5,39e-2	5,46e-2	4,95e-2	8,25e-2
	MSE	5,88e-2	6,16e-2	4,97e-2	8,33e-2
λ_{22}	Sesgo	-0,22	-0,12	-3,42e-3	7,70e-3
	Varianza	4,76e-2	1,13e-2	5,33e-2	4,66e-2
	MSE	9,80e-2	2,49e-2	6,01e-2	4,67e-2
λ_{12}	Sesgo	-9,35e-2	-0,15	1,01e-3	3,62e-2
	Varianza	3,65e-2	2,03e-2	5,31e-3	4,70e-3
	MSE	4,53e-2	4,17e-2	5,43e-3	6,00e-3

Tabla 6.15: Sesgo, varianza y MSE obtenidos al estimar los parámetros para el subgrad. utilizando en la estimación centralizada el grad. conjugado escalado y $\sigma^2_{ruido} = 10^{-2}$.

	Descenso de gradiente				**Gradiente conjugado escalado**			
N	y_0	λ_{11}	λ_{22}	λ_{12}	y_0	λ_{11}	λ_{22}	λ_{12}
100	1,00	1,42	5,01	-1,52	1,00	1,48	5,03	-1,53
200	1,00	1,36	5,01	-1,57	1,00	1,42	4,58	-1,52
500	1,00	1,41	5,01	-1,51	1,00	1,41	5,01	-1,51
1000	1,00	1,47	4,94	-1,51	1,00	1,41	5,00	-1,51

Tabla 6.16: Valor de los parámetros estimados para el método del subgradiente usando el descenso de gradiente y el gradiente conjugado escalado para las estimas locales con $\sigma^2_{ruido} = 0$.

Con el fin de estudiar además la convergencia de la estimación de dichos parámetros, presentamos en la Figura 6.10 un ejemplo de la evolución de dicha convergencia para cada

método, descenso de gradiente (línea verde) y gradiente conjugado escalado (línea azúl), comparado con el valor real (línea roja). Cabe destacar que ambos métodos convergen antes de 100 iteraciones para todos los parámetros, con el método de descenso de gradiente típicamente convergiendo mucho más rápido que el gradiente conjugado escalado.

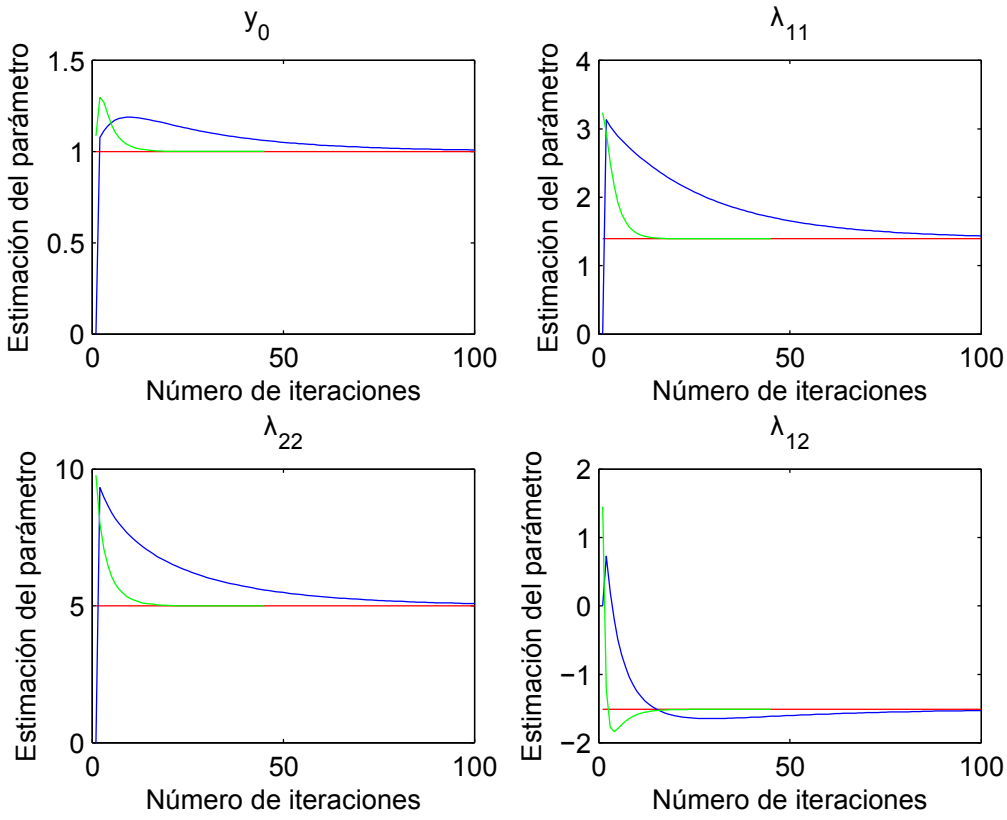
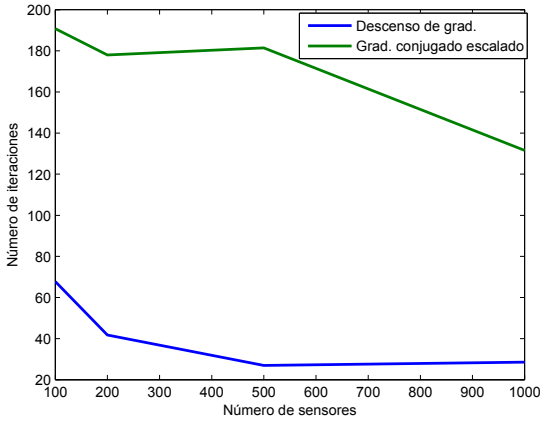


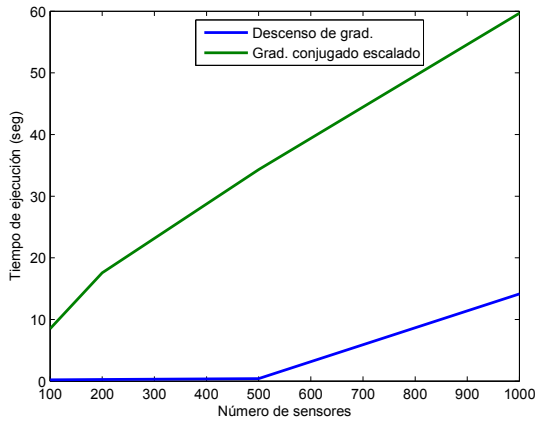
Figura 6.10: Evolución de la convergencia de la estimación de los parámetros con ambos métodos comparado con el real para un valor de $\sigma^2_{ruido} = 0$. En rojo se muestra el valor real de los parámetros, en verde la estimación del descenso de gradiente y en azúl la estimación del gradiente conjugado escalado.

En anteriores ocasiones hemos resaltado el hecho de que la potencia de los nodos de la red es limitada. Por este motivo resulta interesante analizar el tiempo de ejecución y el número de iteraciones necesario para que los algoritmos converjan. En las Figuras 6.11(a) y 6.11(b) se muestran el número de iteraciones y tiempo de ejecución, respectivamente, necesarios para la convergencia del algoritmo del subgradiente con uno de los métodos

de optimización centralizada que estamos examinando. El descenso de gradiente ofrece mejores resultados que el gradiente conjugado escalado, aunque debemos recordar que el método de descenso de gradiente utilizado en las simulaciones es específico para el problema que estamos planteando en este Proyecto Final de Carrera, mientras que para el gradiente conjugado escalado se está utilizando una implementación genérica procedente de la liberería Netlab. Nótese que el número de iteraciones disminuye al aumentar el número de sensores de la red en ambos casos.



(a)



(b)

Figura 6.11: Número de iteraciones y tiempo de ejecución necesario para converger en función del algoritmo local usado para el método del subgradiente y el número de sensores de la red con $\sigma^2_{ruido} = 0$.

6.5. Algoritmo de proyecciones alternas

En esta sección vamos a estudiar el segundo de los algoritmos de estimación distribuida, proyecciones alternas. Al igual que el primero de los algoritmos estudiados, para la implementación del mismo necesitamos la ayuda de un algoritmo iterativo de optimización, de forma que podamos estimar los parámetros con los que poder calcular las medidas de temperatura en cualquier punto del área considerada. En este caso sólo vamos a analizar los resultados obtenidos con el gradiente conjugado escalado, ya que, aunque su convergencia es más lenta, resulta más estable numéricamente que el descenso de gradiente.

Las simulaciones se han realizado para redes con 5 sensores, una varianza de ruido añadida a las medidas sintéticas de 10^{-2} y 10^{-3}, y para 10000 valores distintos de los parámetros iniciales distribuidos uniformemente entre 0 y 10. Como parámetros reales vamos a considerar los que se muestran en las Tablas 6.17 y 6.18. Nótese que en la primera tabla $\lambda_{11} = \lambda_{22}$, mientras que en la segunda $\lambda_{11} \neq \lambda_{22} \neq \lambda_{12} \neq 0$.

$\Delta\, T1\,(^{\circ}\mathrm{C})$	$\Delta\, T2\,(^{\circ}\mathrm{C})$	θ°	$y_0\ (^{\circ}\mathrm{C})$	λ_{11}	λ_{22}	λ_{12}
30	30	0	100	2,85	2,85	0

Tabla 6.17: Primer conjunto de parámetros reales para las simulaciones del algoritmo de proyecciones alternas.

$\Delta\, T1\,(^{\circ}\mathrm{C})$	$\Delta\, T2\,(^{\circ}\mathrm{C})$	θ°	$y_0\ (^{\circ}\mathrm{C})$	λ_{11}	λ_{22}	λ_{12}
40	50	10	100	4,13	5,50	-0,12

Tabla 6.18: Segundo conjunto de parámetros reales para las simulaciones del algoritmo de proyecciones alternas.

La implementación de este algoritmo se hará en base a lo visto en el Capítulo 4, donde presentamos la función de coste, dada por (4.26) en la que consideraremos su término de regularización, $R_f^{(n)}$, igual a 0, de la misma forma que hicimos en la sección anterior, obteniendo las estimas como

$$\hat{\theta}_n = \arg \min_{\theta} \sum_{j \in N_n} \left(f_n \left(\mathbf{x}_n; \hat{\theta}_n \right) - z_{jn} \right)^2 . \tag{6.29}$$

En el Capítulo 4 expusimos que este método, antes de comenzar con la estimación de los parámetros, realiza una búsqueda de los sensores vecinos con los que establecerá comunicación inalámbrica. Esto puede provocar en algunas ocasiones que existan conjuntos de nodos de la red que queden totalmente aislados, es decir, que sólo se comuniquen entre ellos y no intercambien información con el resto de la red, ya sea de manera directa o indirecta (a través de otros nodos). Debido a esta limitación, las simulaciones realizadas para este algoritmo se han realizado con una red de 5 sensores como la de la Figura 6.12, con el fin de asegurar que exista comunicación entre todos los sensores de la red.

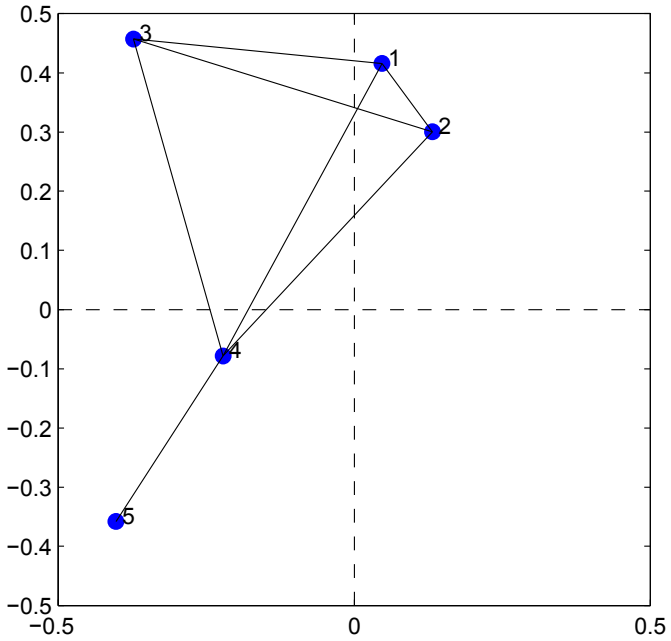


Figura 6.12: Red de 5 sensores distribuidos aleatoriamente sobre un área normalizada entre -0,5 y 0,5 usada para las simulaciones del algoritmo de proyecciones alternas.

A continuación, se muestran los resultados obtenidos en las simulaciones realizadas aplicando el gradiente conjugado escalado al algoritmo de proyecciones alternas, añadiendo cierta varianza de ruido ($\sigma^2 = 10^{-2}$) a los datos sintéticos. Como explicamos anteriormente

en este método cada sensor estima sus propios parámetros a partir de la información que le transmiten sus vecinos. Por tanto, al finalizar las simulaciones tendremos un conjunto de parámetros estimados en cada sensor, aunque idealmente deberían ser todos iguales. En consecuencia, para llegar a un valor de parámetros únicos decidimos calcular la media de los parámetros obtenidos por cada sensor, consiguiendo como resultado final un sólo grupo de parámetros. Teniendo en cuenta las dos tablas de parámetros iniciales definidas al comienzo de esta sección, vamos a tener dos tipos de resultados sobre los que evaluar este método.

- En primer lugar, consideraremos como parámetros reales los mostrados en la Tabla 6.17 y $\sigma^2_{ruido} = 10^{-2}$. En la Tabla 6.19 se muestra el porcentaje de convergencia para cada sensor al estimar cada parámetro de manera independiente. Este cálculo se ha realizado teniendo en cuenta aquellos datos en los que el error de estimación, es decir, la diferencia entre el parámetro estimado y el real es menor que el 10 y el 25 % respectivamente. Para λ_{12}, dicho error es de 10^{-2} y 10^{-1}, respectivamente. En ella se observa el porcentaje de convergencia llega al 100 % para y_0 y λ_{12} y a valores superiores al 99,91 % para λ_{11} y λ_{22}. Además, en la Tabla 6.20 se exponen la media, el sesgo, la varianza y el MSE de los parámetros estimados, considerando un porcentaje de error del 25 %. Cabe destacar el porcentaje de convergencia para el sensor 5 (sensor que sólo está conectado con 4) con un error de estimación del 10 %. Nótese como este valor sube al 95,96 % al considerar un error de estimación del 25 %.

  Los resultados que se mostrarán a continuación son los obtenidos al intentar estimar y_0, λ_{11} y λ_{22} simultáneamente. En la Tabla 6.21 se muestra el porcentaje de convergencia para cada sensor. A diferencia del ejemplo anterior, sólo conseguimos alcanzar casi el 100 % del porcentaje de convergencia en todos los sensores para el parámetro y_0. Por otro lado, la convergencia en el resto de los parámetros rara vez logra alcanzar el 50 %, excepto en casos aislados. Esto demuestra que la convergencia de este algoritmo para el problema planteado en este proyecto no siempre es la deseada. En la Tabla 6.22 se exponen la media, el sesgo, la varianza y el MSE de

Error de estimación (%)	**Sensor**	**1**	**2**	**3**	**4**	**5**
10	y_0	100	100	100	100	100
	λ_{11}	99.91	99.91	99.91	99.91	0
	λ_{22}	99.99	99.99	99.99	99.99	99.99
	λ_{12}	100	100	100	100	100
25	y_0	100	100	100	100	100
	λ_{11}	99.91	99.91	99.91	99.91	95.96
	λ_{22}	99.99	99.99	99.99	99.99	99.99
	λ_{12}	100	100	100	100	100

Tabla 6.19: Porcentaje de convergencia (%) al estimar cada parámetro de forma independiente con el método de proyecciones alternas, tomando como método de estimación centralizada el gradiente conjugado escalado y $\sigma^2_{ruido} = 10^{-2}$.

	y_0	λ_{11}	λ_{22}	λ_{12}
Media	1,00	2,62	2,98	5,3e-2
Sesgo	7,10e-2	-0,23	0,12	5,53e-2
Varianza	1,03e-13	2,85e-4	1,37e-11	8,83e-9
MSE	5,08e-5	7,94e-2	1,48e-2	3,5e-3

Tabla 6.20: Media, sesgo, varianza y MSE de los parámetros estimados para un $\sigma^2 = 10^{-2}$ y un error de estimación del 25 %. Nótese que el valor de y_0 está normalizado.

los parámetros estimados, considerando un porcentaje de error del 25 %.

Por último, en la Figura 6.13 se muestra el error de estimación de la temperatura obtenido al tomar como parámetros de la función Gaussiana los mostrados en la Tabla 6.22 y un valor de $\lambda_{12} = 0$. Como se puede ver, aunque los parámetros estimados no sean los reales el error cometido al estimar la temperatura en cada posición es inferior a 0,03, lo que implica que a pesar de los problemas de convergencia, el algoritmo de proyecciones alternas permite obtener estimas razonables del campo de temperatura.

Error de estimaciòn (%)	Sensor	1	2	3	4	5
10	y_0	94,72	93,56	4,57	94,69	69,11
	λ_{11}	1,40	22,94	0,73	0,00	5,36
	λ_{22}	0,50	32,52	0,41	95,73	29,75
25	y_0	95,35	95,37	95,46	95,43	94,88
	λ_{11}	7,87	50,90	96,55	10,61	13,26
	λ_{22}	1,06	54,55	0,92	97,13	69,15

Tabla 6.21: Porcentaje de convergencia (%) al estimar y_0, λ_{11} y λ_{22} simultáneamente con el método de proyecciones alternas, tomando como método de estimación centralizada el gradiente conjugado escalado y $\sigma^2 = 10^{-2}$.

	y_0	λ_{11}	λ_{22}
Media	1,04	3,00	2,78
Sesgo	4,39e-2	0,15	-7,22e-2
Varianza	1,80e-3	9,12e-2	0.11
MSE	5,7e-3	0,26	0,12

Tabla 6.22: Media, sesgo, varianza y MSE de los parámetros estimados con el algoritmo de proyecciones alternas para un $\sigma^2 = 10^{-2}$ y un error de estimación del 25 % cuando se realiza la estimación conjunta de y_0, λ_{11}, λ_{22} y λ_{12}.

- En segundo y último lugar vamos a tomar como parámetros reales los que se mostraron en la Tabla 6.18, donde $\lambda_{11} \neq \lambda_{22} \neq \lambda_{12} \neq 0$. En la Tabla 6.23 se muestra el porcentaje de convergencia para cada sensor al estimar cada parámetro independientemente del resto. Este cálculo se ha realizado teniendo en cuenta aquellos con un error de estimación menor del 10 y el 25 %, tal y como se indica en la tabla mencionada. Como se puede ver, al introducir un valor distinto de 0 para el parámetro λ_{12} el porcentaje convergencia ha disminuido considerablemente para los parámetros λ_{11} y λ_{12} para un error de estimación del 10 %, mientras que si analizamos los datos obtenidos con un error de estimación del 25 % este efecto se mantiene sólo para el parámetro λ_{12}. En cualquier caso, lo más destacable son los

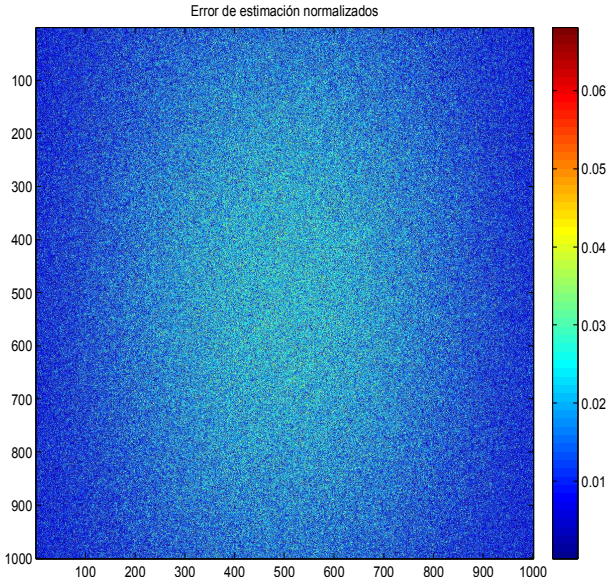


Figura 6.13: Error de estimación con tres parámetros y una varianza de ruido de 0.001.

problemas a la hora de estimar λ_{12}, ya que nunca converge a un valor que esté dentro del margen establecido por el 25 % del error de estimación.

Error de estimaciòn (%)	**Sensor**	**1**	**2**	**3**	**4**	**5**
10	y_0	99,50	99,50	99,50	99,50	99,50
	λ_{11}	0	0	0	0	0
	λ_{22}	100	100	100	100	100
	λ_{12}	0	0	0	0	0
25	y_0	99,98	99,98	99,98	99,98	99,98
	λ_{11}	99,64	99,64	99,64	99,64	99,64
	λ_{22}	100	100	100	100	100
	λ_{12}	0	0	0	0	0

Tabla 6.23: Porcentaje de convergencia(%) al estimar y_0, λ_{11} y λ_{22} de manera independiente con el método de proyecciones alternas, tomando como método de estimación centralizada el gradiente conjugado escalado y $\sigma^2_{ruido} = 10^{-2}$.

Los resultados que se mostrarán a continuación son los obtenidos al intentar estimar la temperatura del foco (y_0), λ_{11} y λ_{22} simultáneamente. En la Tabla 6.24

se muestra el porcentaje de convergencia para cada sensor. Si comparamos con los resultados anteriores podemos destacar que la convergencia para λ_{22} ha disminuido considerablemente para un error de estimación del 10 %.

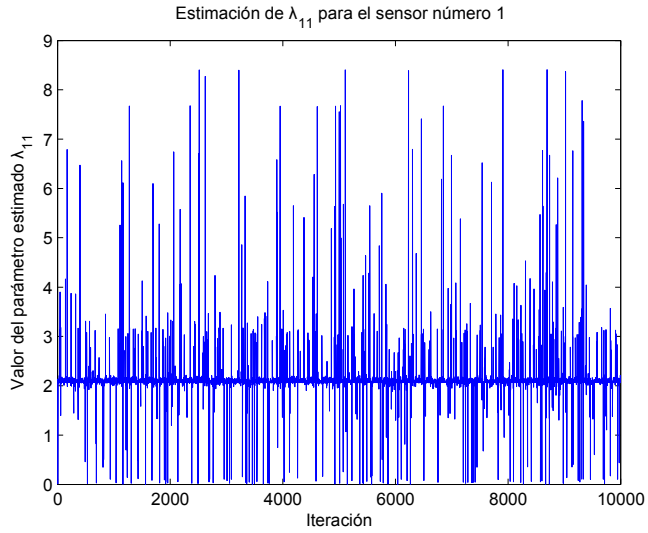
Error de estimaciòn (%)	Sensor	1	2	3	4	5
10	y_0	96,63	80,14	0,14	96,66	95,55
	λ_{11}	1,59	11,02	0,13	0	8,78
	λ_{22}	0,19	18,01	0,03	96,69	58,96
25	Temperatura del foco	96,74	96,92	97,08	96,84	96,73
	λ_{11}	2,29	29,66	97,64	1,43	22,44
	λ_{22}	97,47	42,94	0,51	97,59	98,96

Tabla 6.24: Porcentaje de convergencia(%) al estimar y_0, λ_{11} y λ_{22} simultáneamente con el método de proyecciones alternas, tomando como método de estimación centralizada el gradiente conjugado escalado y $\sigma^2_{ruido} = 10^{-2}$.

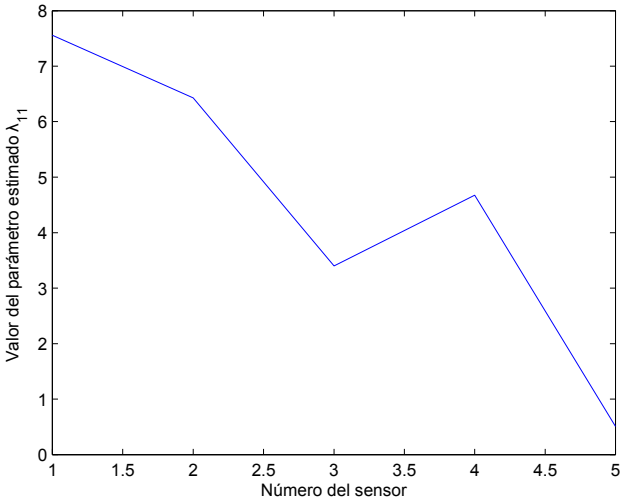
De los resultados obtenidos, podemos destacar que este algoritmo no tiene una convergencia tan buena como la que desearíamos. Para ilustrar un ejemplo de la convergencia del mismo, mostramos las siguientes figuras, donde podremos ver la estimación del parámetro λ_{11} para el sensor 1 (Figura 6.14(a)) y la estimación del parámetro λ_{11} en la simulación 5000 para cada uno de los sensores (Figura 6.14(b)).

6.6. Discusión

En esta sección analizaremos los resultados alcanzados en las diferentes simulaciones realizadas, relacionadas con cada uno de los métodos explicados a lo largo de este Proyecto Final de Carrera. En primer lugar, hemos desarrollado un mecanismo para generar el conjunto de medidas sintéticas necesarias para llevar a cabo el resto de simulaciones. Hemos definido dos formas de originar dichas medidas: directa e indirecta. Nuestros datos sintéticos han sido creados usando la manera indirecta, que conlleva que el usuario únicamente



(a)



(b)

Figura 6.14: (a) Estimación del parámetros λ_{11} para el sensor 1 y (b) estimación del parámetro λ_{11} para cada uno de los sensores en la iteración 5000, considerando en ambos casos el algoritmo de proyecciones alternas.

debe especificar el decremento de temperatura en un área determinada y la inclinación de los ejes. La razón por la cual elegimos este modo se debe a que es más intuitivo definir dichos parámetros, que no determinar qué parámetros (λ_{11}, λ_{22} y λ_{12}) provocarían el descenso de temperatura deseado.

A continuación, una vez fijadas las medidas sintéticas, debíamos averiguar cuál de los métodos definidos en el Capítulo 5 (vecino próximo, inserción e inserción modificado) nos suministraría la ruta óptima a seguir para partir de un nodo y regresar al mismo, después de haber visitado todos los nodos de la red una sola vez, de forma que la distancia recorrida fuera la menor posible. Para ello decidimos comparar la longitud de las distintas rutas proporcionadas por redes formadas por un número variable de sensores, entre 4 y 1000, obtenidas evaluando las rutas óptimas comenzando cada vez por un sensor de la red y, además analizando la longitud de la ruta conseguida al iniciar la búsqueda por un sensor elegido aleatoriamente. En ambos procesos, el camino óptimo lo hemos obtenido con ayuda del método de inserción modificado, desarrollado en este Proyecto Final de Carrera a partir del método de inserción básico. Este método además nos proporciona menor coste computacional, por lo que podríamos considerarlo como el algoritmo más adecuado para nuestro problema.

Para terminar la discusión de este capítulo, deberemos comentar los resultados alcanzados con cada uno de los algoritmos de estimación distribuida:

- Por un lado, tenemos el método del subgradiente cuyas estimaciones se pueden considerar bastante buenas, dado que tanto los parámetros, obtenidos con los dos métodos de optimización centralizada (descenso de gradiente y gradiente conjugado escalado), como la medida del MSE se acercan a lo deseado. Es importante resaltar que el descenso de gradiente proporciona mejores medidas de MSE que el gradiante conjugado escalado. Esto se debe principalmente a que la función de descenso de gradiente utilizada en este proyecto ha sido específicamente implementada para este problema y el parámetro μ elegido es el óptimo para nuestro caso de estudio. Sin embargo, la función de gradiente conjugado escalado utilizada es una función genérica implementada por la librería Netlab.

- Por otro lado, queda por examinar el algoritmo de proyecciones alternas. Como vimos en las simulaciones no hemos podido analizarlo para un número de sensores elevado, debido a que la convergencia de este método no llega a ser buena. Esto se debe a que con este algoritmo cada sensor sólo se comunica con sus vecinos, por

lo que pueden formarse subredes aisladas, haciendo que los parámetros estimados localmente por ese conjunto de nodos no se transmitan nunca al resto de nodos de la red.

Capítulo 7

Conclusiones y Líneas futuras

7.1. Introducción

Una vez expuestos los resultados conseguidos en las distintas simulaciones, resta únicamente presentar las conclusiones obtenidas durante el desarrollo de este Proyecto Final de Carrera, así como las posibles mejoras y líneas futuras a seguir tras la finalización del mismo.

7.2. Conclusiones

El principal propósito de este Proyecto Final de Carrera ha sido la implementación de dos algoritmos de procesado distribuido que, a partir de los datos proporcionados por una red de sensores, fueran capaces de estimar un conjunto de parámetros que nos permitieran calcular una aproximación de la medida real de dichos sensores. Antes de poder llevar a cabo el análisis de estos algoritmos, hemos tenido que realizar otros estudios relacionados con el procesamiento de los datos y la búsqueda del camino óptimo para recorrer la red. A continuación, vamos a exponer cada uno de los temas tratados junto con las conclusiones obtenidas en los mismos.

En primer lugar debíamos plantear un escenario de estudio: una red de N sensores distribuidos aleatoriamente capaces de detectar, procesar y almacenar la información del

medio que les rodea, así como de establecer comunicación inalámbrica con otros terminales cercanos de la red. Entre las principales ventajas de esta clase de redes se encuentran, por un lado, sus numerosas aplicaciones (médicas, militares, de seguridad, medioambientales, etc.), y, por otro lado, el tamaño reducido de los sensores de la red, que nos permite alcanzar sitios inaccesibles hasta el momento y de la forma menos invasiva posible. Como principales inconvenientes, se puede resaltar que aún deben mejorarse algunos aspectos de estas redes relacionados especialmente con la seguridad y el procesamiento eficiente de la gran cantidad de datos que manejan, ya que están muy limitados en capacidad de almacenamiento y consumo de potencia.

En segundo lugar, hemos analizado las ventajas e inconvenientes del procesado distribuido con respecto al centralizado. Atendiendo a las necesidades de nuestras redes y las características de los dispositivos que las forman hemos llegado a la conclusión que, frente al modelo clásico de procesamiento centralizado, estas redes ofrecen múltiples ventajas trabajando con procesado distribuido: menor consumo de potencia y coste, mayor capacidad de cálculo, adaptabilidad, crecimiento, fiabilidad, precisión, mejor aprovechamiento de los recursos de la red, etc. Aún así, presentan ciertos inconvenientes que podrían mejorarse, como por ejemplo el encaminamiento de la información o el rendimiento conseguido por los estimadores distribuidos, que es típicamente inferior al de los centralizados.

En tercer lugar, hemos abordado la implementación y análisis de dos algoritmos de procesado distribuido, subgradiente y proyecciones alternas, para los cuales hemos necesitado la ayuda de métodos empleados también en el procesamiento centralizado: la familia de algoritmos de descenso de gradiente. De las tres técnicas estudiadas en este Proyecto Final de Carrera (descenso de gradiente, gradiente conjugado y gradiente conjugado escalado), hemos concluido que la que mejor se adapta a nuestras necesidades es la del gradiente conjugado escalado, a pesar de alguno de los inconvenientes que presenta. Es el método más complejo de los estudiados, desde el punto de vista de su implementación, aunque por otro lado su éxito no depende de la elección de los parámetros por parte del usuario, como en el descenso de gradiente y gradiente conjugado, y además, en teoría su velocidad de convergencia es mayor que la de los otros dos algoritmos. Con respecto a

esta medida podemos decir que los resultados obtenidos no se ajustan completamente a lo esperado. Esto se debe a que hemos implementado un algoritmo de descenso de gradiente específico para el problema planteado en este proyecto, que converge más rápido que el gradiente conjugado escalado, pero con una formulación de dicho algoritmo de manera más general la velocidad de convergencia disminuye considerablemente. En cuanto a los dos algoritmos de procesado distribuido estudiados (subgradiente y proyecciones alternas) las conclusiones finales se enuncian a continuación:

- Son capaces de llegar a una estimación global de los parámetros a partir de estimas locales de los parámetros en el caso del subgradiente, y a partir de las estimas locales de temperatura de los sensores y promediando todos los parámetros finales de los distintos nodos en el caso de proyecciones alternas.

- El intercambio de información entre sensores necesario para realizar dicha estimación es mínimo: en un caso (subgradiente) únicamente es necesario enviar los parámetros actualizados de un sensor al siguiente de la lista, mientras que en el otro (proyecciones alternas) cada sensor sólo transmite sus observaciones a los nodos vecinos. En ambos métodos esto supone un ahorro importante en consumo de potencia y necesidad de almacenamiento con respecto al modelo centralizado, en el que toda la información debe transmitirse al nodo.

- Con respecto a las simulaciones, por un lado cabe destacar la buena estimación de los parámetros del método del subgradiente, tanto con el descenso de gradiente como con el gradiente conjugado escalado, resaltando que la velocidad de convergencia para el descenso de gradiente programado específicamente para este estudio es superior a la del gradiente conjugado escalado. Por otro lado, nos queda comentar los problemas de convergencia del algoritmo de proyecciones alternas, con el que no hemos obtenido muy buenos resultados de estimación de los parámetros.

En cuarto lugar, como ya hemos comentado, los algoritmos de procesado distribuido necesitan conocer el orden secuencial en el que los nodos se envían la información entre sí, es decir, debemos buscar la ruta óptima a seguir para la transmisión de los datos,

que pase por todos los sensores una sola vez, partiendo y regresando al mismo punto. En este Proyecto Final de Carrera hemos estudiado dos métodos (vecino más cercano e inserción), modificando uno de ellos para posibilitar su implementación eficiente en una red de sensores. En los resultados obtenidos hemos visto que las distancias proporcionadas por cada método no se diferenciaban en gran medida unas de otras. Sin embargo, podemos decir que el que realiza la búsqueda del camino óptimo de la manera más eficiente es el método de inserción modificado, debido a que es algo menos complejo que el método de inserción y por tanto el coste será menor.

Como conclusión final, podemos decir que en una red de sensores inalámbricos distribuidos aleatoriamente sería aconsejable realizar la búsqueda del camino óptimo con el método de inserción modificado y, una vez encontrado, utilizar un procesamiento distribuido sobre la información de la red, de modo que la estimación de los parámetros se realizara con el algoritmo de subgradiente con ayuda del método de gradiente conjugado escalado. Cabe destacar que el método de subgradiente es una buena opción siempre que trabajemos con modelos paramétricos, ya que la optimización de la función se realiza en base a dichos parámetros. Para un modelo no paramétrico podríamos intentar aplicar el algoritmo de proyecciones alternas, aunque antes deberíamos resolver los problemas de convergencia relacionados con el mismo.

7.3. Líneas futuras

Tras la finalización de este proyecto creemos que sería interesante continuar la línea de investigación con los puntos que se enumeran a continuación:

- Implementación y estudio de otros algoritmos de estimación distribuida basados en paso de mensajes, como por ejemplo el algoritmo de entrenamiento distribuido mediante la explotación de dispersión descrito en [4] o los métodos de la familia "Belief Propagation" (BP): BP Generalizado, BP Gaussiano, etc. [63].

- Extensión de los algoritmos aquí estudiados para modelos no paramétricos. En este caso el subgradiente no es implementable, ya que la estimación se hace en base a

una serie de parámetros y en los modelos no paramétricos tendríamos que realizar la estimación de los hiperparámetros asociados. Por otro lado, sería interesante simular y ver el comportamiento del algoritmo de proyecciones alternas para modelos no paramétricos.

- Generalización del mecanismo usado para generar medidas sintéticas para incluir varios focos e incluso estudiar la posibilidad de usar medidas de magnitudes físicas proporcionadas por redes de sensores reales.

- Inclusión de términos de regularización tanto en el método del subgradiente como en el de proyecciones alternas con el fin de mejorar la convergencia en ambos algoritmos (especialmente el de proyecciones alternas) a los parámetros reales.

- Implementación y anális de versiones distribuidas de otros algoritmos para resolver el problema del viajante, como por ejemplo el método de Christofides.

APÉNDICES

APÉNDICE A

Presupuesto del proyecto

Este apéndice está dedicado a la elaboración del presupuesto estimado para la realización del presente proyecto. En la estimación de este importe económico se tiene en cuenta tanto el coste de los medios materiales como el coste de los recursos humanos empleados para su realización.

A.1. Coste de los medios materiales

En la Tabla A.1 realiza un desglose del material utilizado para realizar el proyecto. Al total del coste del material utilizado hay que añadir el coste del emplazamiento de trabajo donde se ha llevado a cabo la realización del proyecto. El lugar de trabajo está debidamente acondicionado mediante alumbrado, calefacción, aire acondicionado, servicio de limpieza, mesas, sillas, tomas de corriente, y red de datos de banda ancha. Su coste en alquiler se estima en unos 750 € por mes con todos los gastos incluidos. Se ha estimado una duración de trabajo del proyecto de 15 meses, por lo que el coste en concepto de alquiler de emplazamiento puede estimarse en 11.250 €.

Concepto	Coste unitario (€)
Ordenador personal	1.000
Licencia de sistema operativo Windows 7	200
Licencia de MATLAB	2.400
Impresora + Cartuchos + Papel	100
Gastos de oficina	100
Encuadernación de los tomos	180
Total material	3.980

Tabla A.1: Coste unitario del material utilizado en la realización del proyecto.

Sumando el coste total del material más el alquiler del lugar de trabajo, se obtiene que el coste total de medios materiales es de 15.230 €.

A.2. Coste del personal

Según el Colegio Oficial de Ingenieros de Telecomunicación (COIT), el sueldo medio de un Ingeniero de Telecomunicación es de 30.000 € brutos anuales. Como referencia, podemos escoger un sueldo inicial para un trabajador en fase de realización del proyecto de fin de carrera o recién titulado de 18.000 € brutos anuales (incluidos beneficios sociales). En dichas condiciones, el salario ronda los 1.500 € brutos por mes, valor que se tomará como referencia. Por lo tanto, el coste de personal del proyecto asciende a un total de 22.500 €.

A.3. Coste de la dirección

Para la realización de este proyecto de ingeniería, estimamos el salario del director, de forma general, como un 7 % de la suma del coste de material más el coste de la mano de obra hasta un límite de 30.000 € anuales. Por tanto el coste de la dirección del proyecto a lo largo de la elaboración de este estudio es de 2.641,1 €.

A.4. Coste total del proyecto

El importe económico final del proyecto se calcula como la suma del coste de los medios materiales más el coste del personal y dirección del proyecto. Realizando este cálculo se obtiene que el coste total del proyecto asciende a 40.371,1 €.

Bibliografía

[1] J. K. Hart and K. Martinez, "Environmental sensor networks: A revolution in the earth system science?" *Earth-Science Reviews*, vol. 78, no. 3-4, pp. 177–191, October 2006.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[3] Nayibe Chio Cho and Diego Alexander Tibaduiza Burgos and Laura Cristina Aparicio Zafra and Luis Miguel Caro Ortiz, "Redes de sensores inalámbricos," in *II Congreso Internacional de Ingeniería Mecatrónica*, vol. 1, no. 1. Universidad Autónoma de Bucaramanga (Colombia), 30 September, 1 and 2 October 2009.

[4] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, July 2006.

[5] S. Boyd and J. Datorro, "Alternating Projections. Notes for course EE364b: Convex optimization II," Autum 2003. [Online]. Available: http://www.stanford.edu/class/ee364b/lectures/alt-proj.pdf

[6] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Regression in sensor networks: Training distributively with alternating projections," in *15th SPIE Conference Advanced Signal Processing Algorithms, Architectures, and Implementations*, San Diego, CA (USA), July/August 2005, pp. 591 006: 1–15.

[7] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, August 2004.

[8] T. Arampatzis, J. Lygeros, and S. Manesis., "A survey of applications of wireless sensors and wireless sensor networks," in *IEEE Proceedings of the 13th Mediterrean Conference on Control and Automation*, Limassol (Chipre), 27-29 June 2005, pp. 719–724.

[9] Cheeyee Chong and Srikanta P. Kumar, "Sensor networks: Evolution, opportunities and challenges," in *Proceedings of the IEEE*, vol. 91, no. 8, 11 August 2003, pp. 1247–1256.

[10] A. Ailamaki, C. Faloutsos, P. S. Fischbeck, M. J. Small, and J. VanBriesen, "An environmental sensor network to determine drinking water quality and security," *SIGMOD Record*, vol. 32, no. 4, pp. 47–52, December 2003.

[11] S. Kumar and D. Shepherd, "SensIT: Sensor information technology for the warfighter," in *Proceedings 4th International Conference on Information Fusion*, vol. 1, Montreal (Canadá), 7-10 August 2001, pp. TuC1: 3–9.

[12] K. Martinez, J. K. Hart, and R. Ong, "Sensor network applications," *Computer*, vol. 37, no. 8, pp. 51–56, August 2004.

[13] A. Matese, S. D. Gennaro, A. Zaldei, L. Genesio, and F. Vaccari, "A wireless sensor network for precision viticulture: The NAV system," *Computers and Electronics in Agriculture*, vol. 69, no. 1, pp. 51–58, August 2009.

[14] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for "smart dust"," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seatle, WA (U.S.A.), August 1999, pp. 271–278.

[15] ITU-R. Terrestrial Services: Frequently Asked Questions. [Online]. Available: http://www.itu.int/ITU-R/terrestrial/faq/index.html

[16] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks." *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 62–74, April/March 2001.

[17] P. A. Boncz, A. Bonifati, J.-H. Böse, S. Böttcher, P. K. Chrysanthis, L. Gruenwald, A. Illarramendi, P. Janacik, B. König-Ries, W. May, A. Mondal, S. Obermeier, A. Ouksel, G. Samaras, B. Sapkota, R. Steinmetz, and S. D. Viglas, "P2P, ad hoc and sensor networks. All the different or all the same?" in *Proceeding on Scalable Data Management in Evolving Networks*, Schloss Dagstuhl (Germany), March 2007, pp. 1–7.

[18] K. Römer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, December 2004.

[19] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, May 2007.

[20] K. A. Delin, S. P. Jackson, D. W. Johnson, S. C. Burleigh, R. R. Woodrow, J. M. McAuley, J. M. Dohm, F. Ip, T. P. Ferré, D. F. Rucker, and V. R. Baker, "Environmental studies with the sensor web: Principles and practice," *Sensors*, vol. 5, no. 1-2, pp. 103–117, February 2005.

[21] C. Council. (2007, March) Cyberinfrastructure vision for 21st century discovery. National Science Foundation. [Online]. Available: http://www.nsf.gov/pubs/2007/nsf0728/index.jsp

[22] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, vol. 284, no. 5, pp. 34–42, May 2001.

[23] W3C. Extensible markup language (XML). [Online]. Available: http://www.w3.org/XML/

[24] Sheng Ye A. and Feng Xuezhi .A and Shaotao Yuan C. and Ruan Renzong D., "GML. An open standard geospatial data format," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, vol. 37, pp. 1773–1778, 2008.

[25] N. Waidyanatha, "Common alerting protocol," in *ITU Asia-Pacific Centre of Excellence Training/Workshop on Effective Use of Telecommunications/ICTs in Response to Disasters: Saving Lives*, Sintok, Kedah (Malaysia), 25 November 2008.

[26] SensorNet. Fibre optic sensors and digital monitoring systems. [Online]. Available: http://www.sensornet.co.uk/

[27] G. Keller, D. Seber, A. Sinha, and C. Baru, "The geosciences network (GEON): One step towards building cyberinfrastructre for the geosciences," 2005. [Online]. Available: http://www.geongrid.org/

[28] Geographic information systems. [Online]. Available: http://www.gis.com/

[29] R. Butler, T. Lay, K. Creager, P. Earl, K. Fischer, J. Gaherty, G. Laske, B. Leith, J. Park, M. Ritzwoller, J. Tromp, and L. Wen, "The global seismographic network surpasses its design goal," *EOS (American Geophysical Union)*, vol. 85, no. 23, pp. 225–232, 8 June 2004.

[30] Center for the built environment. [Online]. Available: http://www.cbe.berkeley.edu/

[31] W. Bourgeois, P. Hogben, A. Pike, and R. M. Stuetz, "Development of a sensor array based measurement system for continuous monitoring of water and wastewater," *Sensors and Actuators B: Chemical*, vol. 88, no. 3, pp. 312–319, August 2003.

[32] National Oceanic and Atmospheric Administration. (NOAA). Earth as a new frontier: The world-changing capability of the global earth observation system of systems (geoss). [Online]. Available: http://www.noaa.gov/eos.html

[33] J. K. Hart, K. C. Rose, K. Martinez, and R. Ong, "Subglacial clast behaviour and its implication for till fabric development: New results derived from wireless subglacial

probe experiments," *Quaternary Science Reviews*, vol. 28, no. 7-8, pp. 597–607, April 2009.

[34] K. Martinez, P. Padhy, A. Riddoch, H. Ong, and J. Hart, "Glacial environment monitoring using sensor networks," in *Proceedings of the 1st Workshop on Real-World Wireless Sensor Networks*, Stockholm (Sweden), 20-21 June 2005.

[35] André Lins and Eduardo F. Nakamura and Antonio A.F. Loureiro and Claudionor J.N. Coelho Jr., "Beanwatcher: A tool to generate multimedia monitoring applications for wireless sensor networks," *Lecture Notes in Computer Science*, vol. 2839, pp. 128–141, 2003.

[36] A. Bharathidasan and V. A. S. Ponduru, "Sensor networks: An overview," Department of Computer Science, University of California, Davis, Tech. Rep., 2002.

[37] D. Niculescu, "Positioning in ad hoc sensor networks," *IEEE Network*, vol. 18, no. 4, pp. 24–29, July/August 2004.

[38] A. Silberschatz, *Operating system concepts*. John & Wiley Sons, 2005.

[39] C. J. Stone, "Consistent nonparametric regression," *The Annals of Statistics*, vol. 4, no. 4, pp. 595–645, July 1977.

[40] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Consistency in models for distributed learning under communication constraints," *IEEE Transactions on Information Theory*, vol. 52, pp. 52–63, January 2006.

[41] D. Remondo, "Tutorial on wireless ad hoc networks," in *International Working Conference in Performance Modelling and Evaluation of Heterogeneous Networks (HET-NET)*, Ilkley, West Yorkshire (U.K.), 26-18 July 2004, pp. 1–15.

[42] J. B. Escario, C. D. Martín, and V. N. López, "Encaminamiento adaptativo en redes ad hoc mediante el algoritmo ant colony optimization," Ph.D. dissertation, Universidad Complutense de Madrid, 2008.

[43] D. Balfanz, D. Smetters, P. Stewart, and H. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS)*, Palo Alto, CA (U.S.A), February 2002.

[44] M. Lázaro, M. Sánchez-Fernández, and A. Artés-Rodríguez, "Optimal sensor selection in binary heterogeneous sensor networks," *IEEE Transactions on Signal Processing*, vol. 57, no. 4, pp. 1577–1587, April 2009.

[45] S. S. Petrova and A. D. Solov'ev, "The origin of the method of steepest descent," *Historia Mathematica*, vol. 24, no. 4, pp. 361–375, November 1997.

[46] S. S. Haykin, *Adaptive filter theory.* Upper Saddle River, New Jersey (USA): Prentice Hall, 2005.

[47] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," School of Computer Science Carnegie Mellon University, Pittsburgh, PA (U.S.A.), Tech. Rep., 4 August 1994.

[48] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, December 1952.

[49] R. Fletcher and M.Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[50] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, Ed. Springer, 2000.

[51] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, 1993.

[52] R. Fletcher, *Practical Methods of Optimization.* John Wiley & Sons, New York, NY (USA), 1975.

[53] M. R. Hestenes, "Conjugate direction methods in optimization," *Lecture Notes in Control and Information Sciences*, vol. 6, pp. 8–27, 1978.

[54] N. Z. Shor, N. G. Zhurbenko, A. P. Likhovid, and P. I. Stetsyuk, "Algorithms of nondifferentiable optimization: Development and application," *Journal of Cybernetics and Systems Analysis*, vol. 39, no. 4, pp. 537–548, 2003.

[55] J. von Neumann, *Functional Operators, Volume II: The Geometry of Orthogonal Spaces.*, A. of Math. Studies, Ed. Princeton University Press, 1950, no. 22.

[56] S. Sahni and T. González, "P-Complete approximation problems," *Journal ol the Association for Computing Machinery*, vol. 23, no. 3, pp. 555–565, July 1976.

[57] Von einem alten Commis-Voyageur [Un viejo comerciante], *Der Handlungsreisende - wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein.* B.Fr. Voigt, 1832.

[58] W. R. Hamilton and T. Kirkman, *Graph Theory 1736-1936.* Oxford (UK): Oxford University Press, 1976.

[59] A. Schrijver, "On the history of combinatorial optimization (till 1960)," in *Handbooks in Operations Research and Management*, K. Aardal, G. Nemhauser, and R. Weismantel, Eds. Amsterdam (The Netherlands): Elsevier, September 2005.

[60] A. N. Kolmogorov y S. V. Fomin, *Introductory Real Analysis*, New York, NY (USA), 1970.

[61] Daniel J. Rosenkrantz and Richard E. Stearns and Philp M. Lewis II, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no. 5, pp. 563–581, 1977.

[62] I. Nabney, *NETLAB : algorithms for pattern recognition.* Springer, 2004.

[63] J. Yedida, W. Freeman, and Y. Weiss, *Exploring artificial intelligence in the new millennium*, G. Lakemeyer and B. Nebel, Eds. San Francisco, CA (USA): Morgan Kaufmann, January 2003.