UNIVERSIDAD AUTÓNOMA METROPOLITANA

Localización en Redes Inalámbricas de Sensores

Idónea Comunicación de Resultados para obtener
el grado de

MAESTRO EN CIENCIAS

por

Carlos Ernesto Moreno Escobar

Asesores

Dr. Ricardo Marcelín Jiménez
Dr. Miguel Ángel Ruiz Sánchez

Sinodales

Presidente: Dr. Jesús Figueroa Nazuno
Secretario: Dr. Ricardo Marcelín Jiménez
Vocal: Dr. Michael Pascoe Chalke
Vocal: Dr. Enrique Rodríguez de la Colina

21 de enero de 2011

Resumen

Una red inalámbrica de sensores WSN (*Wireless Sensor Networks*) es esencialmente un conjunto de nodos sensores con alimentación propia que recolectan información y se comunican de forma inalámbrica, con una meta común.

Las WSN son una tecnología emergente con un amplio espectro de aplicaciones potenciales como el monitoreo de la salud humana, vigilancia, control de incendios, observación de la fauna, entre otras. No obstante, estas redes enfrentan retos difíciles, como asignar a cada nodo una posición de manera precisa. En esto consiste el problema de la localización. La localización de los nodos es una herramienta importante que permite crear nuevas aplicaciones para las WSN. Por ejemplo, los sistemas de monitoreo pueden identificar el origen de un evento crítico. También, la información que provee la localización puede mejorar el funcionamiento de las WSN. Por ejemplo, los nodos pueden enviar paquetes a su destino final, basándose únicamente en la posición de los nodos en su vecindario. Esta estrategia de encaminamiento fomenta el trabajo local y limita el consumo de energía.

Para resolver el problema de la localización, un sistema de posicionamiento global (GPS) puede ser un punto de partida, pero el uso del GPS es actualmente una solución costosa que no puede usarse en interiores, donde la recepción satelital es pobre. Para un conjunto pequeño de nodos, las posiciones individuales pueden programarse manualmente. No obstante, si se requiere emplear un número masivo de sensores, entonces la configuración manual de la posición no es una opción. En algunos casos se emplean nodos móviles que conocen su posición en todo momento y realizan un recorrido sobre la red, informando a los nodos su ubicación. Sin embargo, si la red se despliega aleatoriamente o un nodo móvil no es factible, entonces se requiere un procedimiento automático que realice la localización.

En este trabajo se propone abordar el problema de localización en una red de sensores inalámbricos. Se considera que los nodos se encuentran en posiciones fijas, pero desconocidas. Igualmente, se asume que los nodos no disponen de ningún circuito complementario para estimar rangos o distancias. El método presentado se desarrolla en cuatro etapas sucesivas: en la primera, la WSN se particiona. En la segunda, por cada una de las colonias resultantes se calcula la distancia en saltos entre cada pareja de nodos de la misma colonia. En la tercera, se resuelve localmente el problema de escalamiento multidimensional. Finalmente, se introduce un conjunto de faros en cada partición, para ensamblar los fragmentos de la red en un sistema de coordenadas globales.

Aquí se ofrece evidencia experimental que sugiere que la partición no sólo se reduce el costo computacional y de comunicaciones, sino que también permite trabajar sobre redes con topología y dimensiones arbitrarias. Por otro lado, se reconocen cuáles son los parámetros críticos que influyen

en la solución, tales como la densidad de la red o el tamaño de sus particiones. En el primero, se encontró que la incertidumbre de la solución es inversamente proporcional a la densidad de la red. Por otro lado, esta misma incertidumbre es directamente proporcional al tamaño de la partición.

La partición aparece como una condición necesaria para escalar los algoritmos de localización. La red se divide en zonas más pequeñas, cada una de las cuales queda a cargo de una versión reducida del problema original, pero sobre una subgráfica con una densidad más uniforme y una topología más regular. Esto facilita una organización basada en recursos locales, con la capacidad de coordinarse en un contexto global.

También se analizó el desempeño de tres algoritmos que resuelven el escalamiento multidimensional (MDS Clásico, SMACOF y la combinación de ambos) herramienta fundamental de este trabajo. Identificamos cuál de ellos es el algoritmo conveniente para resolver la localización, basándonos en su complejidad computacional y la calidad de los resultados que produce.

Agradecimientos

Agradezco a la UAMI y al CONACyT por el apoyo otorgado para la realización de este trabajo. A las personas que me han acompañado y apoyado durante esta etapa de mi vida maravillosa y plena de buenas experiencias. Les agradezco a todos mis maestros —*me hayan impartido cátedra o no*— sus enseñanzas, sus valiosos comentarios tanto profesionales como personales. A toda mi familia: tíos y tías, mis abuelos, primos y primas y a mis siempre queridos sobrinos. A mis padres, Agus y Chayo, que espero no decepcionarlos, les agradezco con todo mi amor, con la promesa de que nunca me voy a rendir, nunca voy a retroceder a pesar de los obstáculos que se interpongan en mi camino, gracias, los amo. A mis hermanos, Ramón y Gaby, que siempre fueron ejemplo de vida y yo, siempre me resistí a verlo. Su incondicional apoyo ha hecho de mi lo que soy, me han hecho ver más allá, buscar más... querer más.

Angelina y Marcelino: ¿qué cosa fuera la maza sin cantera? Sus conocimientos y experiencias hicieron un vivo de mí, capaz de ver la vida de una manera sencilla, práctica... única. A mi compadre y mejor amigo Carlos Alberto. Alejandra Victoria, mi Matemática, te amo, ¿qué sería el álgebra lineal sin ti?.

Al Dr. Ricardo Marcelín, espero que esté satisfecho con este trabajo y que dé pauta a un trabajo más exhaustivo. Sus consejos y sabiduría no se quedarán en el aire, se lo prometo. México necesita más personas como usted.

Finalmente, con todo mi amor, a Salvador Moreno Basurto. Tú mueves el mundo.

Ing. Carlos Ernesto Moreno Escobar
Diciembre de 2010

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos y Procedimientos

Acrónimos

AoA Angle of Arrival

BFS Breadth-First Search

CPU Central Processing Unit

CBLALS Cluster Based Three-Dimensional Localization Algorithm for Large Scale Wireless Sensor Networks

DES Discrete Event Simulator

DFS Depth-First Search

DV Distance Vector

DV-Hop Distance Vector Hop

ELA Elastic Localization Algorithm

GPS Global Positioning System

MA-MDS-MAP(P) Mobile Asisted Multi Dimensional Scaling Map using Patches

UAM-MDS UAM Multidimensional Scaling

MDS-MAP Multi Dimensional Scaling Map

MDS-MAP(P) Multi Dimensional Scaling Map Partition using Patches

RF Radio Frecuencia

RSSI Received Signal Strenght Indicator

SMACOF Scaling by MAjorizing a COmplicated Function

SVD Singular Value Decomposition

SSS Simulation Subroutine Set

TDoA Time Diference of Arrival

ToA Time of Arrival

UDG Unit Disk Graph

WSN Wireless Sensor Network

Capítulo 1
Introducción

Las redes inalámbricas de sensores WSN (Wireless Sensor Networks) son una tecnología emergente que puede emplearse en una amplia gama de aplicaciones [2]. Estas redes se conforman por un conjunto de dispositivos electrónicos llamados nodos. Cada nodo está compuesto por una unidad central de proceso (CPU), memoria, dispositivos de radiofrecuencia *-por ejemplo, un módulo Bluetooth*. Sin embargo, el elemento principal que incluye el nodo es un dispositivo electrónico capaz de monitorear alguna variable física dentro de su área de operación. Este dispositivo es un sensor, el cual, es capaz de interactuar con el ambiente para medir distintos parámetros físicos en el mundo real, por ejemplo, la temperatura, humedad, movimiento, vibraciones, contaminación, etc.

Las características mencionadas anteriormente dan pauta a una gran variedad de aplicaciones para una WSN. Algunas aplicaciones que utilizan este tipo de redes son el monitoreo de la salud humana, edificios inteligentes, vigilancia, prevención de desastres, control de incendios, observación de la fauna, agricultura, entre otras.

Una red de sensores se compone de un gran número de nodos que pueden estar densamente desplegados dentro de la zona de observación. Además, la posición de los nodos no necesariamente es determinada previamente, esto implica que el posicionamiento de la red puede realizarse de manera aleatoria en terrenos inaccesibles o zonas de desastre, sobre todo si no se tiene conocimiento *a priori* de la zona de despliegue.

La localización se ha convertido en una operación fundamental para el correcto funcionamiento de las WSN ya que algunos servicios son soportados por el conocimiento de la posición de los nodos que provee la localización. Desde un punto de vista de la aplicación, los usuarios pueden reconocer la fuente de un evento crítico basados en la posición del transmisor. Sin embargo, la información que provee la localización puede emplearse también en niveles operacionales. Por ejemplo, es posible transmitir un paquete a su destino final empleando únicamente la posición de los nodos en su vecindario. Esta estrategia de encaminamiento permite reducir el consumo de energía [19].

La solución trivial para que un nodo estime su posición es muy simple: grabar la posición del nodo manualmente, es decir, programar el microcontrolador del nodo con algún tipo de identificador que sirva como referencia para conocer el lugar en que el nodo se ha posicionado. Sin embargo, si se requiere emplear un número masivo de sensores, entonces la configuración manual de la posición no es una opción. Un sistema de posicionamiento global (GPS) puede emplearse para estimar la posición de los nodos. Sin embargo, el uso del GPS es actualmente una solución costosa que funciona únicamente en exteriores.

1.1. Definición del Problema

En muchas circunstancias es útil o incluso necesario que un nodo en una red inalámbrica de sensores estime su posición geográfica. Por ejemplo, el rastreo o la detección de algún evento no son particularmente útiles si la red de sensores no puede proporcionar la información de *dónde* ha ocurrido un evento.

La localización es el proceso de estimar las posiciones de los nodos a partir de la medición de las distancias entre los mismos. Las mediciones pueden clasificarse en dos categorías: esquemas basados en rango (*Range-Based*) y esquemas libres de rango (*Range-Free*). Los algoritmos basados en rango miden la distancia mediante hardware especializado para estimar sus posiciones. Por otro lado, los esquemas libres de rango se basan enteramente en la información de conectividad, por ejemplo, *quién está en el radio de alcance de quién.*

Una red puede modelarse por la gráfica $G = (V, E)$, donde $v \in V$ es un nodo y $e \in E$ indica que existe una comunicación directa entre un par de nodos. En la mayoría de los casos, una WSN se modela como una *gráfica de disco unitario* UDG (*Unit Disk Graph*). En una UDG, $G = (V, E)$ y hay una arista $\{u, v\} \in E$ si y solo si la distancia Euclidiana entre u y v menor o igual que 1.

El *empotrado* de una gráfica $G = (V, E)$ en el plano Euclidiano es una transformación $f : V \to \mathbb{R}^2$, donde cada vértice v_j, $j = 1, 2, \ldots, n$ es identificado como un punto $x_j \in \mathbb{R}^2$ en el plano. Una *realización* de UDG $G = (V, E)$ en el plano Euclidiano, es un empotrado de G tal que $\{u, v\} \in E \leftrightarrow d(f(v), f(u)) \leqslant 1$, donde d es la distancia Euclidiana entre dos puntos. De esta manera, la localización consiste en la realización de una gráfica de disco unitario en el plano Euclidiano [21].

La localización también se puede entender como un problema de optimización. Dado un conjunto de distancias entre los nodos que conforman la red, se requiere estimar las posiciones de cada nodo en el plano, salvo rotación y traslación, tal que, el error entre las distancias estimadas y las distancias reales se minimiza. Generalmente se asume que una pequeña parte de los nodos de la red, llamados *faros*, tienen conocimiento previo de sus coordenadas. Estas coordenadas se obtienen colocando los faros en posiciones conocidas o mediante el uso de un GPS, por ejemplo. Así, los faros se emplean para construir el sistema global de coordenadas de la red de sensores o asistir el proceso de localización (véase la Sección 2.1.1).

Matemáticamente, un nodo normal [1] j cuya posición será estimada se denota como $x_j \in \mathbb{R}^2$, $j = 1, 2, ..., n$. En contraste, cada nodo faro k, cuya posición es conocida se denota como $a_k \in \mathbb{R}^2, k = 1, 2, ..., m$. Sea d_{ij} la distancia Euclidiana entre cada par de nodos normales i y j, y sea d_{jk} la distancia Euclidiana entre un nodo normal j y un nodo faro k.

En muchos casos, no todos los pares de distancias son conocidas, así, los pares de nodos cuyas distancias son conocidas son denotadas como $(i, j) \in N_x$ para el par sensor-sensor y $(j, k) \in N_a$ para el par sensor-faro.

El problema de la localización en $\mathbb{R}^2$ puede establecerse como: dadas m posiciones de nodos faro $a_k, k = 1, 2, ..., m$ y algunas mediciones de distancias $d_{ij}, (i, j) \in N_x, d_{jk}, (j, k) \in N_a$, encontrar x_j, $j = 1, 2, ..., n$, las posiciones de los nodos normales tal que

[1] Los nodos normales son los nodos que aún no estiman su posición. También suele llamárseles *nodos desconocidos*

$$|x_i - x_j|^2 = d_{ij}^2, \forall (i,j) \in N_x \quad (1.1)$$
$$|x_j - x_k|^2 = d_{jk}^2, \forall (j,k) \in N_a \quad (1.2)$$

En algunas circunstancias, las mediciones ruidosas de distancias introducen incertidumbres en los cálculos. Entonces, el problema puede reformularse de la siguiente manera

$$\text{mín}\{|x_i - x_j|^2 - d_{ij}\} \quad (1.3)$$
$$\text{mín}\{|x_j - x_k|^2 - d_{jk}\} \quad (1.4)$$

1.2. Justificación

En una red de sensores, la localización es una tarea importante que involucra la colaboración de todos los nodos dentro de la red. La localización es un servicio fundamental debido a que muchas aplicaciones de este tipo de redes requieren el conocimiento de la posición de sus integrantes, incluso si la red está compuesta por cientos o miles de ellos. Además, la información que provee la localización puede emplearse también en niveles operacionales como el encaminamiento.

El problema de la localización requiere una gran cantidad de herramientas donde los diseñadores deben encontrar el método más apropiado, dependiendo de la instancia particular del problema. Los métodos con complejidades polinomiales funcionan bien sobre redes homogéneas. Sin embargo, estos métodos deben reemplazarse o adaptarse cuando las mediciones son ruidosas. Además, el problema se vuelve NP-Completo cuando se trabaja con redes ralas, entonces, las heurísticas ofrecen soluciones limitadas pero con una complejidad razonable.

Se han observado nuevas tendencias después de identificar las dificultades del problema. La partición aparece como una condición necesaria para escalar los algoritmos de localización. La red se divide en zonas más pequeñas, cada una de las cuales queda a cargo de una versión reducida del problema original, pero sobre una subgráfica con una densidad más uniforme y una topología más regular. Esto facilita una organización basada en recursos locales, con la capacidad de coordinarse en un contexto global.

1.3. Objetivos

1.3.1. Objetivo General

En este proyecto se propone el desarrollo de un protocolo de localización distribuido para estimar la posición de los nodos en una red inalámbrica de sensores, de tamaño y topología arbitrarios, posicionados en un plano de dos dimensiones usando únicamente la información de conectividad.

El escenario de implementación propuesto para el desarrollo de este proyecto es el siguiente:

1. Los sensores se encuentran en posiciones fijas pero desconocidas.
2. El tamaño y la densidad de la red es arbitrario.
3. El área de despliegue de la red es un plano de dos dimensiones con topologías arbitrarias (regulares y con obstáculos).
4. Se considerará el uso de nodos faro.

1.3.2. Objetivos Particulares

- Evaluar las soluciones existentes
- Reconocer cuáles son los parámetros críticos que influyen en la calidad de la solución (como cantidad de nodos faro, densidad de la red, tamaño, etc)
- Elegir o proponer una solución de acuerdo con una lista de requerimientos

1.4. Metodología

1. Revisión de la literatura en el tema
2. Creación del protocolo de investigación
3. Caracterización de los requerimientos del sistema
4. Selección de una herramienta de simulación
5. Implantación
6. Evaluación de casos de estudio
7. Comunicación de resultados

1.5. Contribución

En este trabajo se propone abordar el problema de localización en una red de sensores inalámbricos, de tamaño y topología arbitrarios. Se considera que los nodos se encuentran en posiciones fijas, pero desconocidas. Igualmente, se asume que los nodos no disponen de ningún circuito complementario para estimar rangos o distancias. El método presentado se desarrolla en cuatro etapas sucesivas: en la primera, la WSN se particiona. En la segunda, por cada una de las colonias resultantes se calcula la distancia en saltos entre cada pareja de nodos de la misma colonia. En la tercera, se resuelve localmente el problema de escalamiento multidimensional. Finalmente, se introduce un conjunto de faros en cada partición, para ensamblar los fragmentos de la red en un sistema de coordenadas globales.

Este no es el primer trabajo que emplea un enfoque de partición para resolver el problema de la localización. La mayoría de las contribuciones particionan la red excesivamente de tal forma que cada nodo que subyace en la red es *líder* de colonia, el cual es el encargado de coordinar las operaciones en ella. Este comportamiento incrementa el costo computacional y de comunicaciones, puesto que cada nodo es miembro de varias colonias simultáneamente y por tanto, participa en distintas operaciones innecesariamente. En cambio, el algoritmo de particionamiento a emplear en este trabajo, crea las colonias con un diámetro que puede parametrizarse para acotar los costos del computacionales. Por otro lado, sólo el líder realiza cálculos en su colonia, esto permite reducir costos de ejecución del algoritmo.

Aquí se ofrece evidencia experimental que sugiere que la partición no sólo se reduce el costo computacional y de comunicaciones, sino que también permite trabajar sobre redes con topología y dimensiones arbitrarias. Por otro lado, se reconocen cuáles son los parámetros críticos que influyen en la solución, tales como la densidad de la red o el tamaño de sus particiones. En el primero, se encontró que la incertidumbre de la solución es inversamente proporcional a la densidad de la red. Por otro lado, esta misma incertidumbre es directamente proporcional al tamaño de la partición.

También se analizó el desempeño de tres algoritmos que resuelven el escalamiento multidimensional, herramienta fundamental de este trabajo. Identificamos cuál de ellos es el algoritmo conveniente para resolver la localización, basándonos en su complejidad computacional y la calidad de los resultados que produce.

Capítulo 2
Estado del Arte

Los algoritmos de localización se dividen en dos categorías: esquemas basados en rango y los esquemas libres de rango. Estos esquemas difieren en la manera en que se obtienen las distancias. Los esquemas basados en rango se basan en mediciones de distancias físicas o ángulos entre los nodos de la red mediante hardware especializado. Los esquemas libres de rango consideran únicamente la información de conectividad entre nodos adyacentes. Estos esquemas emplean algoritmos para eliminar la necesidad de medir distancias, por esta razón, no requieren hardware adicional.

De acuerdo con el lugar en que los cálculos se realizan los algoritmos de localización se clasifican en algoritmos centralizados y algoritmos distribuidos. En los primeros todas las mediciones, al igual que los cálculos para estimar las posiciones de los nodos, las realiza un sólo nodo. En los segundos cada nodo participa de manera cooperativa para medir distancias y estimar su propia posición.

En la mayoría de los algoritmos, tanto en los basados en rango como en los libres de rango, se considera que existe una pequeña porción de nodos que tiene conocimiento *a priori* de sus posiciones, tales sensores son llamados *faros* o *anclas* y se emplean tanto para construir sistemas de coordenadas globales, como para fungir como puntos de referencia para que los nodos, aún no localizados, se auxilien en el proceso.

Este capítulo se organiza como sigue: en la primera parte se presentan los conceptos fundamentales involucrados en la localización de las redes inalámbricas de sensores (WSN), así como las clasificaciones de los algoritmos de localización. También se presentan las técnicas comúnmente empleadas para medir distancias entre dos sensores. Finalmente, en la segunda parte de capítulo se presentan las contribuciones más renombradas en la localización y su comparación.

2.1. Definiciones

Los esquemas basados en rango miden las distancias físicas o ángulos entre los nodos de la red. Sin embargo, estos esquemas requieren la instalación de hardware adicional para estimar las distancias, lo que incrementa el costo y consumo energético de los nodos. Los esquemas libres de rango consideran únicamente la información de conectividad entre nodos adyacentes. Estos esquemas emplean algoritmos para eliminar la necesidad de medir distancias, por esta razón, no se requiere hardware adicional. Por otro lado, ambos esquemas pueden emplear algunos nodos con posición conocida para auxiliarse en la localización. Estos nodos son llamados *faros* (también conocidos

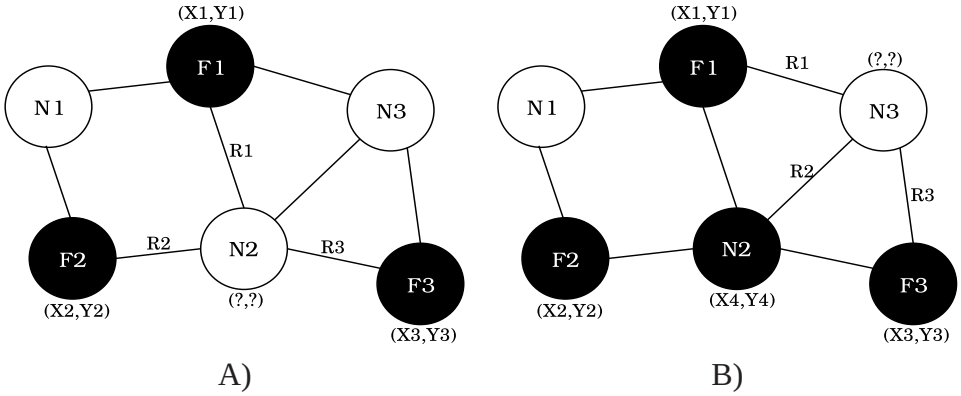


Figura 2.1: A) El nodo desconocido *N2* estima su posición ayudándose de los nodos Faro $F1$,$F2$ y $F3$. B) Una vez que el nodo *N2* estima su posición es elevado a Faro, así, puede auxiliar al nodo *N3* a estimar su propia posición.

como Anclas o Semillas). Para algunos algoritmos de localización, el funcionamiento de los nodos faro puede describirse como sigue: los nodos que aún no estiman su posición (llamados *desconocidos* o *normales*) estiman la distancia hacia los faros y entonces los utilizan como auxiliares para estimar su propia posición, por ejemplo, realizan una triangulación con los nodos faro. Para otros, los faros se emplean únicamente para construir un sistema global de coordenadas para toda la red.

2.1.1. Faros

En la localización de sensores de redes inalámbricas, generalmente se asume que un pequeño número de sensores tienen conocimiento *a priori* de sus propias posiciones. Estos nodos son llamados faros y tienen dos tareas fundamentales en el proceso de localización. Por un lado, permiten construir un sistema de coordenadas común para toda la red, esto es, todos los nodos hacen referencia al mismo origen coordenado como $(0, 0)$, por ejemplo. Los nodos faro, por otro lado, también se emplean como auxiliares en la localización de los nodos desconocidos. Los nodos faro son sólo una pequeña porción del total de nodos dentro de la red de sensores. Por ejemplo en [22, 31] se requiere que el 5 % o el 10 % del total de nodos sean faros para obtener una solución con el mínimo error. En estas contribuciones, los faros asisten el proceso de localización donde los nodos desconocidos estiman las distancias hacia los faros y entonces, mediante algún tipo de cálculo, como una triangulación por ejemplo, estiman su propia posición.

Algunas contribuciones pueden utilizar los faros con el enfoque siguiente: considérese la Figura 2.1.A) donde hay tres faros que conocen su posición, estos son: $F1$, $F2$ y $F3$. Ahora, cada uno de los faros le transmite al nodo $N2$ su propia posición. Cuando el nodo $N2$ conoce, tanto las posiciones de los tres faros como las distancias a ellos ($R1$, $R2$ y $R3$) entonces, empleando geometría elemental, $N2$ puede estimar su propia posición. Esto sugiere que los algoritmos de localización pueden requerir un mínimo de faros para lograr una localización precisa.

Para sobrellevar el problema de las insuficiencias de faros es posible emplear un enfoque recursivo, esto es, cuando un nodo desconocido ha estimado sus coordenadas puede elevarse a la

condición de faro. Esto hace posible que los nodos desconocidos que están lejos de los faros iniciales sean capaces de localizarse con los faros convertidos. Esta situación se ilustra en la figura 2.1B), donde el nodo $N2$ ha estimado su posición con ayuda de los faros a su alrededor, así, este nodo asciende a la condición de faro y es capaz de auxiliar al nodo $N3$, junto con los faros $F1$ y $F3$, para que éste estime su posición. Un problema de los enfoques recursivos es que el error de localización puede acumularse, distorsionando el resultado final. Sin embargo, en la mayoría de la soluciones, los nodos pueden convertirse en faro cuando la incertidumbre en su posición es mínima.

En las contribuciones donde los faros asisten la localización se requiere el conocimiento *a priori* de las posiciones de los faros. Esto implica que el número de faros que se necesitan depende del algoritmo con el que se aborda el problema. Sin embargo, algunas otras contribuciones emplean a los faros únicamente para construir las posiciones globales de la red, esto es, los faros no tienen ninguna participación durante la ejecución del algoritmo y por tanto permiten reducir considerablemente la cantidad de faros requerida. En [28, 27], por ejemplo, sólo se necesitan 3 faros para ubicar en $2D$ o 4 para $3D$.

Los faros no se consideran hardware adicional, sino una herramienta auxiliar para complementar los algoritmos de localización, independientemente de qué esquema se use: basados en rango o libres de rango. Cabe mencionar que algunos autores consideran que los faros tienen algunas características adicionales, tales como recursos energéticos superiores al resto de los nodos o un rango de comunicaciones más extenso. No obstante, en este trabajo la única característica principal de los faros es que en todo momento tienen conocimiento de sus posiciones.

Finalmente, las coordenadas de los nodos faro pueden obtenerse empleando un sistema de posicionamiento global (GPS) o instalándolos en posiciones con coordenadas conocidas. Sin embargo, la instalación de un receptor GPS en cada nodo es una solución costosa. Además, el GPS no puede emplearse en interiores, donde la recepción satelital es pobre.

2.1.2. Esquemas Basados en Rango

Los llamados *esquemas basados en rango* (Range-Based) utilizan distintas técnicas para la medición de distancias y dependen, esencialmente, del intercambio de señales entre dos nodos. La recepción de la señal puede ser función de la distancia entre los nodos, de la posición relativa y la línea de vista entre el transmisor y el receptor. En estas técnicas se requiere un modelo de propagación de la señal electromagnética; hardware especializado basado en señales acústicas o, incluso, arreglos de antenas para estimar el ángulo de arribo de la señal de RF [9, 15, 17, 18].

Indicador de la Potencia de la Señal Recibida *RSSI*

El RSSI (*Received Signal Strength Indicator*) es una técnica basada en la potencia medida de la señal electromagnética. La potencia de la señal recibida por un sensor es una función monótona decreciente de la distancia [18]. La relación de la potencia de la señal recibida y la distancia es, generalmente, modelada como:

$$P_r(d)[dBm] = P_0(d_0)[dBm] - 10n_p \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{2.1}$$

Donde $P_0(d_0)[dBm]$ es la potencia del transmisor, n_p es el exponente de pérdida de la trayectoria, el cual, mide la tasa a la cual decrece la potencia de la señal recibida cuando la distancia se incrementa. X_σ es una variable aleatoria Gaussiana de media 0 y desviación estándar σ, la cual, representa el efecto aleatorio causado por el desvanecimiento. Tanto n_p y σ dependen del medio ambiente, por lo que la precisión de esta medición depende de la técnica particular para construir el modelo y el algoritmo usado para ajustar la medición de la señal al modelo. Sin embargo, emplear el RSSI en interiores es complicado ya que es difícil determinar los parámetros del modelo. Cuando se usa el RSSI para la medición de distancias es necesario aceptar y trabajar bajo considerables errores de medición [9, 25].

Algunas propuestas optan por el uso o caracterización del RSSI [25, 26] ya que está integrado en la mayoría del hardware de comunicaciones, así, dado el modelo y sus parámetros, las distancias inter-sensores pueden estimarse bajo experimentación.

Tiempo de Arribo y Diferencia de Tiempo de Arribo

El tiempo de arribo ToA (*Time of Arrival*) y la diferencia de tiempo de arribo TDoA (*Time Diference of Arrival*) son técnicas de medición de la distancia entre dos nodos que emplea el tiempo de propagación de la señal. El ToA estima la distancia con base en el tiempo que demora la señal en llegar del transmisor al receptor, por ejemplo, un nodo envía una señal a sus vecinos a una velocidad predeterminada y posteriormente, espera una respuesta. Dependiendo del tipo de señal empleada para estimar la distancia, como una señal de RF, el tiempo de arribo requiere de relojes con una resolución alta para producir una precisión aceptable. Por esto, es recomendable emplear una señal distinta a la de RF, como un pulso de ultrasonido, por ejemplo. Así, la distancia entre los nodos se estima empleando tanto los tiempos de transmisión y recepción como la velocidad de propagación del pulso en el aire.

Emplear una señal distinta a la de RF, como el pulso de ultrasonido, implica que las resoluciones de los relojes no son tan exigentes, sin embargo, la velocidad del sonido depende de factores externos tales como la temperatura o la humedad.

Por otro lado, el TDoA estima la distancia midiendo la diferencia de propagación de una señal de radio y una acústica generadas en el mismo punto. El transmisor envía una señal de radiofrecuencia seguida de una señal acústica. Cada nodo que escucha las señales de audio mide la diferencia de tiempo de llegada de ambas señales y estima la distancia entre los dos puntos.

Ángulo de Arribo

Las mediciones AoA (*Angle of Arrival*) pueden realizarse mediante dos técnicas: aquellas que hacen uso de la respuesta en amplitud de la antena del receptor y aquellas que se obtienen usando la respuesta en fase de la antena del receptor. En la primer categoría, las mediciones del ángulo están basadas en el patrón de recepción de la antena. Entonces, si la antena es capaz de rotar sobre su eje, la dirección de la máxima potencia de la señal es tomada como la dirección del transmisor. La precisión de este tipo de mediciones depende, en gran parte, de la sensibilidad de la antena. Sin embargo, el uso de antenas giratorias puede ser inapropiado para una red inalámbrica de sensores [15].

La otra técnica empleada para medir el ángulo de arribo consiste en las mediciones de las diferencias de fase durante el arribo del frente de onda. Este tipo de mediciones requieren que el receptor cuente con una antena de mayor tamaño (en comparación con la longitud de onda de la señal del transmisor) o un arreglo de antenas. Sin embargo, ambas técnicas están limitadas por multitrayectorias y direccionalidad de la antena.

2.1.3. Esquemas Libres de Rango

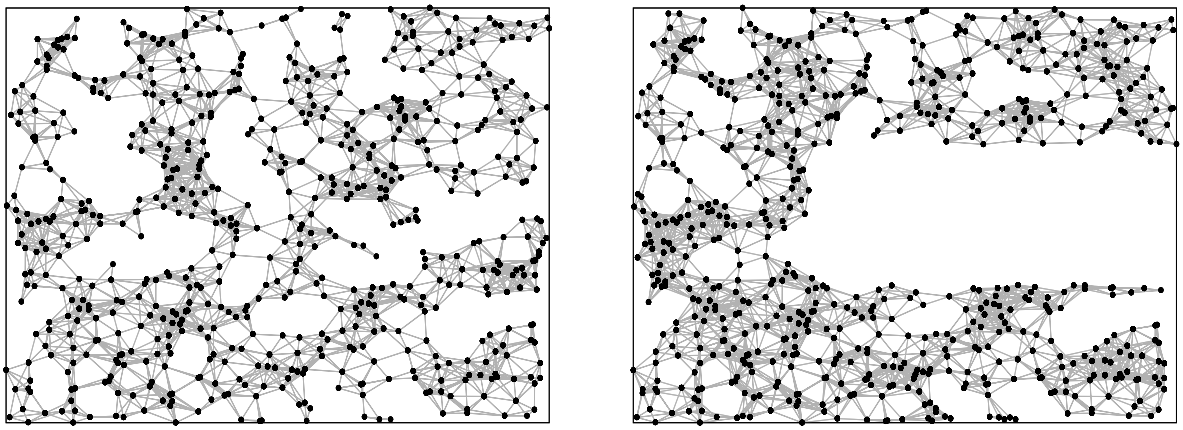
En los esquemas libres de rango, los algoritmos se basan en la información de conectividad, por ejemplo: *quién está en el radio de alcance de quién* y no en la medición de la distancia para estimar la posición de los nodos. La conectividad puede medirse identificando *quienes* son los vecinos de cierto nodo. El principio de este tipo de algoritmos es: *si dos nodos sensores son capaces de comunicarse entre ellos, entonces hay una gran probabilidad de que la distancia entre los nodos sea menor o igual al radio de comunicación*. La ventaja principal de estos esquemas es que no se requiere hardware adicional. Debido a que la localización de los nodos no requiere mediciones de distancia, el empleo de un esquema de rango libre implica simplicidad y bajo costo.

El hecho de que los esquemas libres de rango no cuenten con hardware especializado para medir la distancia inter-nodos no es una limitante para que estos estimen su propio rango, es decir, en los esquemas libres de rango la distancia inter-nodos se mide en saltos, sin embargo, es posible aproximar la distancia entre nodos con distintas técnicas. Por ejemplo, en algunas propuestas como [22], la primera etapa emplea el algoritmo de vector distancia (DV) para estimar la mejor ruta entre un nodo normal y un nodo faro, posteriormente, los faros aproximan la longitud (en metros) de la trayectoria de un nodo normal a un faro. Posteriormente, los nodos normales estiman su posición mediante triangulación. Por otro lado en [31], la distancia entre cada par de nodos se estima mediante el promedio del número de nodos sensores acumulados a lo largo de la trayectoria para ir de un nodo a otro, así, conociendo el radio de alcance de la señal de RF, es posible estimar la distancia física entre dos nodos.

2.2. Aproximaciones

La desventaja de los esquemas basados en rango es que requieren hardware adicional que puede tener un impacto considerable en el precio individual de los nodos. Además, las técnicas de medición de la distancia pueden ser sensibles a las condiciones ambientales. En contraste, las técnicas libres de rango dependen del número de saltos que separan a cualquier pareja de nodos. En este caso, se asume que estos dos nodos comparten una arista siempre que puedan comunicarse uno con otro de manera directa. En ambas categorías, las mediciones se completan empleando procedimientos distribuidos como el algoritmo del vector distancia (DV) donde cada nodo propaga las distancias que conoce a diferentes puntos de la red.

Los algoritmos pueden dividirse dependiendo de dónde se realizan los cálculos. En los métodos centralizados, todas las mediciones se realizan en el punto donde el modelo global fue construido. Cuando se resuelve el modelo, cada nodo recibe la información con su posición estimada. Sin embargo, la desventaja de los métodos centralizados es tanto la complejidad de las operaciones a



A) B)

Figura 2.2: A) Red con topología “uniforme”, B) red con topología irregular *forma-c*.

realizar y la cantidad de mensajes transmitidos. Por ejemplo, un algoritmo centralizado requiere intercambiar una gran cantidad de mensajes para que toda la información concerniente a cada nodo dentro de la red sea llevada a un sólo punto. Así, la complejidad en comunicaciones puede impactar en el consumo de energía de todo el sistema y finalmente, en su tiempo de vida. En los métodos distribuidos, por otro lado, cada nodo intercambia mensajes únicamente con sus vecinos para estimar su posición, además, el empleo de una arquitectura distribuida implica una colaboración de todos los nodos dentro de la red, esto es, las operaciones que requiere la localización pueden realizarse por todos los nodos, esta metodología no es posible en una arquitectura centralizada.

Adicionalmente, muchos algoritmos requieren la medición de distancias o ángulos entre un conjunto de faros y los nodos normales o incluso, emplean un arreglo de faros con posiciones conocidas [1, 14, 22, 25, 26, 29].

Otro tipo de propuestas [7, 28, 29, 30, 32] emplean como base una técnica de análisis de datos llamada *Escalamiento Multi-Dimensional* (MultiDimensional Scaling)[1] que muestra la estructura de distancia entre cada par de nodos como una *fotografía geométrica*. En otras palabras, la entrada del sistema MDS no es más que una matriz de distancias entre cada par de nodos dentro de la red. Así, mediante la *eigendescomposición* de esta matriz, la salida del sistema es una matriz de coordenadas de las posiciones relativas de cada nodo. Comparado con otras soluciones, el MDS puede alcanzar resultados muy precisos pero a un costo computacional que depende de la técnica para realizar la descomposición espectral de la matriz de distancias.

Niculescu y Nath [22], presentan el algoritmo libre de rango DV-Hop, el cual es similar a los algoritmos de encaminamiento basados en el *Vector-Distancia* (DV) [11]. En la primera etapa, se emplea el algoritmo DV para que cada nodo normal estime el mínimo número de saltos hacia los faros. Posteriormente, los faros estiman la longitud promedio de un salto y es transmitida como una corrección mediante un flujo controlado hacia toda la red. Una vez que los nodos normales reciben

[1]El MDS es una herramienta que no es exclusiva de los esquemas libres de rango, de hecho, muchas otras contribuciones que emplean el MDS asumen que es posible estimar la distancia entre cada par de sensores.

la corrección pueden estimar la distancia, en metros, hacia cada uno de sus faros y así, mediante una triangulación estimar su propia posición. Este paso corresponde a la tercera y última etapa del algoritmo. La longitud del salto puede estimarse como

$$c_i = \frac{\sum \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\sum h_i}, \quad i \neq j \tag{2.2}$$

donde (X_i, Y_i) y (X_j, Y_j) son las coordenadas del faro i y el faro j, respectivamente. h_i es el número de saltos para ir del faro i al faro j.

Es evidente que el empleo de una técnica que estima la longitud promedio de un salto en la red, es una herramienta adecuada para un algoritmo libre de rango. Sin embargo, la desventaja de este esquema, es que trabaja sólo sobre topologías uniformes y sin obstáculos, es decir, cuando las propiedades de la gráfica son las mismas en todas direcciones (Figura 2.2A) y B)). Por otro lado, una ventaja importante es su simplicidad y además no depende del rango de error de medición.

Shang et al. [28], presentan el método MDS-MAP para calcular las posiciones de los nodos empleando únicamente la información de conectividad disponible, esto es, quienes están en el radio de comunicaciones de quién. Este método tiene tres pasos básicos: calcular las trayectorias más cortas entre cada par de nodos, arreglándolas como una matriz. Para esto, puede emplearse un algoritmo Bellman-Ford o Dijkstra [10]. Entonces, emplean el escalamiento multi-dimensional (MDS) para estimar las posiciones de los nodos que se ajusten a las distancias estimadas. Finalmente, se obtiene un mapa absoluto tomando en cuenta los faros cuyas posiciones son conocidas.

El MDS reconstruye las posiciones tanto en dos como en tres dimensiones y además, no se requiere un conocimiento *a priori* de las posiciones de los nodos faro hasta la tercera etapa. La desventaja de este método es que es centralizado, es decir, un sólo nodo realiza todos los cálculos necesarios para generar la salida MDS. Por ejemplo, el MDS requiere la descomposición espectral de la matriz de distancias, entonces, si la red está compuesta por un número masivo de nodos un sólo nodo debe realizar dicha descomposición sobre una matriz grande.

Dado que MDS-MAP emplea la trayectoria más corta entre dos nodos para aproximar la distancia Euclidiana, este estimado es apropiado sólo si la red es densa y uniforme pero no lo es para redes irregulares. En [27] se presenta el algoritmo MDS-MAP(P) el cual es capaz de trabajar con esta problemática. En MDS-MAP(P), cada nodo calcula el MDS para crear un mapa local con sus vecinos más cercanos (aquellos que estén a lo más n saltos de distancia). Los mapas locales se mezclan basándose en los nodos que tienen en común, de acuerdo con la mejor transformación lineal que mezcla las coordenadas de los nodos de un mapa a otro. Los pasos del algoritmo son los siguientes: establecer el diámetro de los mapas locales ($1, 2, \ldots, n$ saltos) y calcular las distancias más cortas entre cada par de nodos dentro del vecindario. Cada nodo realiza el MDS para crear un mapa local con los nodos dentro del vecindario. Así, cada mapa local se mezcla uno con otro basándose en los nodos que tienen en común cada mapa para crear un mapa global. Finalmente, con tres o cuatro nodos faro, convertir el mapa global en un mapa absoluto. Este método construye un mapa relativo para cada nodo y su vecindario, por esto, la información local es mucho más precisa y además, la construcción del mapa global con esta técnica es mucho más eficiente para topologías irregulares. Sin embargo, el inconveniente de esta propuesta es que cada nodo que compone la red debe calcular un MDS con respecto a él y a sus vecinos, esto sugiere que el MDS se ejecuta varias

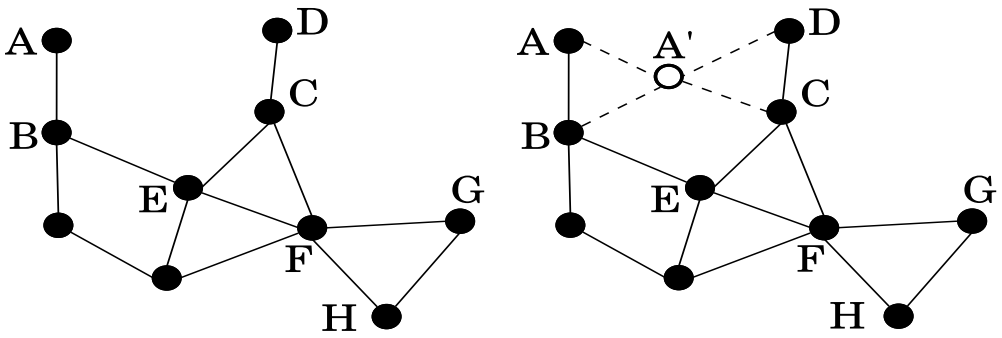


Figura 2.3: A) Gráfica. B) Con el nodo virtual A' la gráfica se hace más rígida.

veces de manera innecesaria.

Vicaire y Stankovic [31], con el algoritmo de localización elástica ELA (Elastic Localization Algorithm), discuten el uso del modelo físico de oscilaciones y relajaciones de un resorte. Inicialmente, tratan de encontrar la gráfica empotrada que se asemeje lo más posible la configuración real. Posteriormente, modifican el empotrado empleando un método de optimización llamado *masa-resorte* para corregir y balancear los errores. El método masa-resorte es una técnica de optimización distribuida, donde, para una red $G = (V, E)$, cada arista $e = (u, v) \in E$, se considera un resorte que une dos masas u y v. Se asume que la longitud del resorte, en estado de equilibrio, es igual a la distancia que existe entre dos nodos. Si la distancia estimada es menor que la distancia medida, el resorte empuja ambos nodos en dirección contraria. Por otro lado, si la distancia estimada es mayor que la distancia medida, el resorte tira de las masas para que estas se acerquen. Este procedimiento es iterativo. En cada paso, los nodos son colocados de acuerdo a las fuerzas que actúan sobre ellos. El proceso finaliza cuando cada nodo alcanza el equilibrio, es decir, cuando la fuerza resultante sobre toda la red es cero.

Wu et al. [32]. Presentan la problemática que existe en el algoritmo MDS-MAP y sus variantes como MDS-MAP(P). Estos métodos suponen que la distancia entre dos nodos es proporcional a la distancia Euclidiana. Esto no necesariamente es cierto si la red es rala o irregular. Por ejemplo, en la Figura 2.3A) se observa que la trayectoria entre el punto A y G es aproximadamente proporcional a la distancia Euclidiana. Por otro lado, la trayectoria entre A y D es significativamente mayor que su distancia Euclidiana. También, otro problema más con el MDS-MAP es que requiere que la red sea lo más rígida posible. Por ejemplo, consideremos nuevamente la Figura 2.3A). Si se calcula el MDS en esta gráfica, entonces la reconstrucción del mapa relativo no sería similar a la gráfica original debido a que la rigidez de la gráfica es mínima en los puntos A, B y C, D, esto implica que las posiciones relativas de los nodos A y D podrían resultar en un lugar completamente arbitrario, aún satisfaciendo las condiciones del MDS.

Entonces, MA-MDS-MAP(P) es un algoritmo que soluciona estas problemáticas empleando asistencia móvil, es decir, emplean un nodo que se mueve a través de la red de sensores creando *nodos virtuales* que fortalecen la gráfica. En la Figura 2.3B), el nodo virtual A' se creó con el nodo móvil, nótese que la gráfica se hace más rígida, lo cual implica que la reconstrucción MDS va

a arrojar un mapa relativo similar al original. Una vez que se generan todos los nodos virtuales virtuales necesarios, entonces se ejecuta el algoritmo MDS-MAP(P) para localizar los sensores.

Shu et al. [30], propusieron CBLALS, el cual, es un algoritmo basado en rango que emplea el MDS para la localización. A diferencia de [32, 28, 7] los cuales, son centralizados, en [30] se realiza el MDS de una manera distribuida. Esto es, empleando un método de partición, la red se divide en varias subredes, llamadas *colonias*. En cada colonia, se ejecutan los procedimientos básicos del MDS de manera independiente, es decir, en cada una se ejecuta un algoritmo de *ruta más corta*, para crear la matriz de distancias y, posteriormente, realizar la descomposición espectral de esta matriz para obtener las posiciones relativas de todos los elementos que componen la colonia. Una vez realizado este procedimiento en todas las subredes, todos los mapas relativos correspondientes a cada colonia se mezclan para crear un sólo mapa global. Entonces, con ayuda de los faros, el mapa global se convierte en un mapa global absoluto. A diferencia de MDS-MAP(P), CBLALS emplea un algoritmo de partición en cual sólo hay un salto de distancia del *líder de colonia* a todos los integrantes de la colonia, además, sólo los líderes ejecutan el MDS para obtener el mapa relativo de la colonia. En MDS-MAP(P), por el contrario, todos los nodos de la red ejecutan el MDS para obtener un mapa relativo de él y sus vecinos más cercanos. Lo que implica que el MDS se ejecuta un número masivo de veces.

La desventaja de esta contribución es que es completamente basada en rango, inclusive, el algoritmo de partición requiere la medición de la distancia entre dos nodos vecinos. Por otro lado, la partición de una red permite no sólo reducir los costos de comunicación sino también simplificar los cálculos necesarios para las operaciones requeridas por el MDS.

En la Tabla 2.1 se resume las características principales de los algoritmos de localización considerados aquí.

Algoritmo Características	DV-HOP [22]	MDS-MAP [28]	MDS-MAP(P)[27]	ELA [31]	MA-MDS-MAP(P)[32]	CBLALS [30]
Complejidad	-	SVD	SVD	-	SVD	SVD
Centralizado Distribuido	Dist	Cent	Dist	Dist	Dist	Dist
Basado en Rango- Libre de Rango	Libre	Libre	Libre	Libre	Basado	Basado
Número de Faros	$\leq 5\,\%$	3 o 4	3 o 4	$\leq 5\,\%$	3 o 4	3 o 4
Conocimiento *a priori* de los faros	Si	No	No	Si	No	No
Particiones	No	No	Si	No	Si	Si
Asistencia móvil	No	No	No	No	Si	No
Obstáculos e irregularidades	No	Si	Si	Si	Si	Si

Notación:
SVD Pasos necesarios para realizar la eigendescomposición.

Tabla 2.1: Características Principales de los algoritmos

Capítulo 3
Herramientas y Métodos

En este capítulo se presentan cuatro herramientas fundamentales para el algoritmo de localización propuesto. En la primera parte se presenta el algoritmo de partición de Awerbuch, el cual divide la red en pequeñas subredes, llamadas colonias. También se muestra una variante del algoritmo de partición capaz de acelerar el proceso de partición. En la segunda parte, se estudia el algoritmo Bellman-Ford, el cual proporciona las rutas más cortas entre una pareja arbitraria de nodos. Este algoritmo produce matriz de distancias a partir de la matriz de adyacencias de una red (o colonia). Finalmente, en la tercera y última parte se presenta la herramienta llamada Escalamiento Multi-Dimensional (MDS), la cual es una técnica matemática empleada en psicología para estimar las posiciones de los nodos en un espacio Euclidiano de dos o tres dimensiones. En esta parte también se sintetizan todos los principios relacionados con el MDS así como tres algoritmos que lo resuelven: MDS Clásico, SMACOF y la combinación de ambos algoritmos.

3.1. Algoritmo de Partición de Awerbuch

3.1.1. Sincronizador γ

En [3] se presenta el sincronizador γ, cuyo objetivo es emular un algoritmo síncrono sobre una red asíncrona. Esto es, permitir al usuario escribir un algoritmo como si fuera a ejecutarse en una red síncrona. Es una metodología simple para diseñar algoritmos distribuidos eficientes en redes asíncronas. Para realizar esta tarea, el sincronizador genera *pulsos de reloj* en cada nodo de la red satisfaciendo la siguiente propiedad: un nuevo pulso es generado en un nodo si y sólo si se reciben todos los mensajes del algoritmo síncrono enviado a un nodo por sus vecinos en pulsos previos. Este comportamiento asegura que la red se comporta de manera síncrona desde el punto de vista de una ejecución particular del algoritmo síncrono.

Más específicamente, dado un algoritmo π_S, diseñado para una red síncrona y un sincronizador ν, es posible combinar π_S y ν para obtener un protocolo $\pi_A = \nu(\pi_S)$ que pueda ejecutarse en una red asíncrona. Así, la emulación debe ser correcta en el sentido de que la ejecución de π_A en una red asíncrona debería ser similar a la ejecución de π_S en una red síncrona [23].

El sincronizador γ necesita una etapa de inicialización, en la cual, la red se particiona en *colonias*. La partición se define por un bosque abarcador de la gráfica $G = (V, E)$ de la red. Cada árbol del bosque define una partición de nodos. Dentro de cada partición, un *líder* se selecciona para

coordinar las operaciones de cada partición. Luego de inicializarse, el sincronizador γ opera en dos fases. En la primera fase, un nodo s se se elige como *líder* de colonia y genera el primer pulso de ejecución. Cuando finaliza la propagación del pulso, el líder eventualmente identifica que todos los nodos de la colonia se encuentran asegurados[1]. Siempre que un líder identifica que su partición está asegurada, reporta este hecho a los nodos miembros de su partición como a los líderes de las particiones vecinas. Ahora, los nodos que coordina entran a la segunda fase, en la cual esperan hasta que todas las particiones vecinas se encuentren aseguradas y generen el siguiente pulso del algoritmo síncrono.

3.1.2. Particionamiento

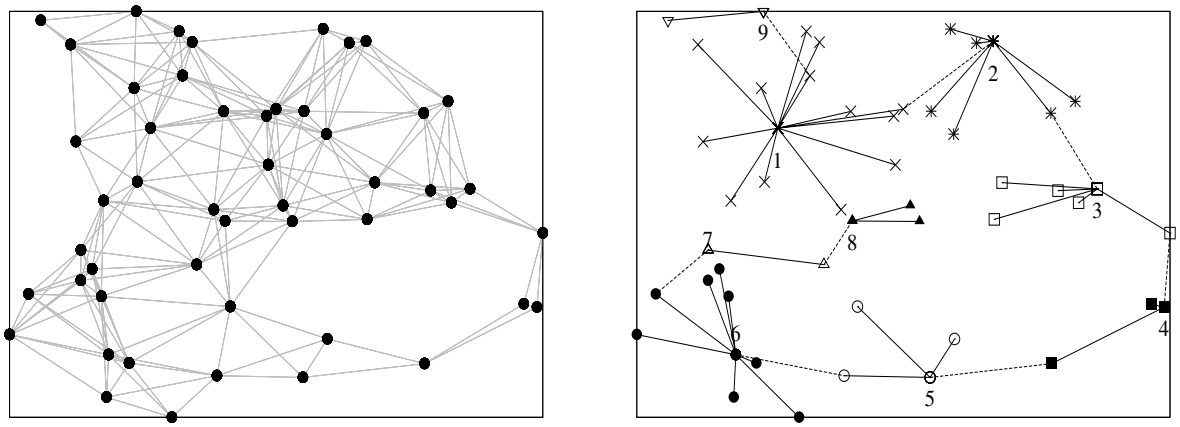
La columna vertebral del sincronizador γ es el algoritmo de partición. La idea principal de este algoritmo es seleccionar una colonia como un subconjunto máximo de nodos cuyo diámetro no exceda el logaritmo de su cardinalidad. De esta forma se garantiza que el número total de colonias vecinas sea lineal y el diámetro máximo de cada colonia sea igual al logaritmo del número de nodos de la red.

La etapa de inicialización procede construyendo las colonias una por una, de tal manera que un nodo seleccionado de la gráfica residual (la subred que no se ha unido a ninguna colonia) y la colonia esté formada a partir de dicho nodo. Este procedimiento continúa hasta que no quedan más nodos en la gráfica residual. El algoritmo opera como sigue. Un nodo seleccionado como líder inicia la ejecución de un algoritmo de búsqueda en amplitud sobre la gráfica residual. Cada nueva capa BFS se une a la colonia hasta que el número de nodos en cierta capa sea menor que $k-1$ veces el número total de nodos contenidos en todas las capas previas (k se denomina el *factor de crecimiento de las colonias*), así, la colonia generada forma un árbol BFS con raíz en el líder.

Básicamente, el algoritmo de partición es sólo un algoritmo BFS distribuido, el cual, construye el árbol capa tras capa. Al inicio del pulso número P, $P-1$ capas del árbol ya han sido construidas. El propósito del pulso P es unir o rechazar la capa P para finalizar el procedimiento de creación de la colonia. La decisión de unir la capa P depende del número total de nodos que hay en esta capa.

El líder l transmite un mensaje $PULSE$ sobre el árbol existente, cada nodo interno en la capa $P' < P-1$ propaga el mensaje $PULSE$ recibido de su padre hasta alcanzar la capa $P-1$. Una vez que se recibe este mensaje en la última capa $P-1$, el nodo i propaga el mensaje $LAYER\{P-1, l\}$ a todos sus vecinos, informándoles que él pertenece a la capa $P-1$, cuyo líder de colonia es l. Una vez que el vecino j que no pertenece a ninguna colonia recibe este mensaje, se une a la capa P del líder l. Entonces j envía un mensaje $ACK\{bit\}$ al nodo i con $bit = 1$ si se ha unido a la colonia y $bit = 0$ en otro caso. Para calcular el número de nuevos nodos que se han unido en la capa P, cada nodo espera hasta que el número de descendientes en la capa P es conocido, entonces, reporta este número a su padre mediante un mensaje $COUNT\{*\}$. Un nodo en la capa $P-1$ sabe el número de nodos nuevos que se unieron a la colonia mediante los mensajes ACK recibidos de sus vecinos. El procedimiento termina cuando el líder conoce el número total de nodos en la capa P. Si el número

[1]Se dice que un nodo o una colonia están asegurados respecto a cierto pulso de ejecución si cada mensaje del algoritmo síncrono enviado por un nodo —*o una colonia* —en ese pulso ya ha arribado a su destino. Después de la ejecución de un cierto pulso, eventualmente un nodo o una colonia estarán asegurados.



A) B)

Figura 3.1: A) Red Inalámbrica de Sensores (Gráfica residual). B) Red inalámbrica de sensores después de la ejecución del algoritmo de partición Awerbuch.

de nuevos elementos agregados a la colonia es al menos $k - 1$ veces mayor que el número total de nodos en la colonia, entonces el líder inicia el siguiente pulso $P + 1$. En otro caso, transmite a lo largo del árbol el mensaje $REJECT$, el cual notifica a los nodos pertenecientes a la capa P que son ahora expulsados de la colonia, esto es, la relación "padre-hijo"entre los nodos de la capa $P - 1$ y P se cancela.

Una vez que cierta colonia C se crea, la capa rechazada se examina. Si no está vacía, entonces un nodo en esta capa se selecciona como un nuevo líder. En el caso en que la capa esté vacía, el centro de la actividad retrocede (*backtrack*) hasta la colonia desde la cual C fue descubierta y el procedimiento arriba descrito se repite ahí. Este procedimiento de formación de colonias se realiza en forma secuencial.

Entre dos colonias vecinas se selecciona un *enlace preferido* que funciona como canal de comunicaciones entre esas colonias. Para seleccionar el enlace preferido, se le asignan distintos pesos a todos los enlaces. El peso del enlace (i, j) es el par $(\text{mín}(i, j), \text{máx}(i, j))$. Así, el enlace preferido seleccionado es aquel enlace de menor peso[2]. Esta regla permite al líder de colonia seleccionar un sólo enlace de todos los enlaces incidentes sobre una misma colonia. La elección del enlace preferido se desarrolla una vez que el centro de la actividad retrocede para continuar creando las colonias.

Para entender de una manera sencilla el procedimiento de partición consideremos la Figura 3.1A), la cual es una red compuesta por 50 nodos. El algoritmo de partición es ejecutado dando como resultado la Figura 3.1B). En esta figura, se observa claramente que la red se particiona en nueve colonias, donde cada una se crea secuencialmente. Esto es, la primer colonia creada mediante una búsqueda BFS es aquella identificada con el número 1. Después, se selecciona la siguiente colonia que ha de ser creada: la colonia 2. Este procedimiento se repite y termina hasta que la

[2]Awerbuch propone seleccionar como enlace preferido al enlace de menor peso de todos los posibles. Sin embargo, si este criterio es cambiado no se altera funcionamiento del algoritmo.

gráfica residual esté vacía, es decir, todos los nodos se han incorporado a una colonia.

3.1.3. Complejidad del Algoritmo de Particionamiento

Sea E_P el conjunto de los enlaces que participan en el sincronizador, es decir, los enlaces de los árboles interiores y los enlaces preferidos para una partición P. Sea H_P la altura máxima de un árbol en el bosque P. En concreto, el algoritmo de partición optimiza las propiedades E_P y H_P de la gráfica $G = (V, E)$. Así, se produce la subgráfica tal que $|E_P| < k|V|$ y $H_P < \log_2 V/\log_2 k$, para un parámetro k, $2 \leq k \leq |V|$. La elección del factor del crecimiento de las colonias k relaciona un compromiso entre el tiempo y el número de mensajes. Entonces, si cada colonia se construye según el procedimiento descrito anteriormente, se tiene

$$\begin{aligned} H_P \leq \log_k |V| = \frac{\log_2 |V|}{\log_2 k} \\ |E_P| \leq k|V| \end{aligned} \tag{3.1}$$

La fase de inicialización, durante la cual se designa al líder de la primera colonia, se completa empleando un algoritmo de elección cuya complejidad es $C_{MST} = O(|E| + |V| \log_2 |V|) = O(|V|^2)$ y $T_{MST} = O(|V| \log_2 |V|)$.

Por cada colonia que resulta de la partición, el procedimiento de construcción se invoca exactamente una vez y éste toma $\log_k |V|$ pulsos. Los mensajes que se intercambian durante este procedimiento pueden clasificarse en 2 tipos: los que se envían en cada etapa y los que se envían una sola vez exactamente. Al primero corresponden los mensajes $PULSE$ y $COUNT$, los cuales atraviesan cada arista del árbol interior, en cada etapa de la construcción. Al segundo corresponden los mensajes $LAYER$ y ACK que se intercambian entre los nodos del último nivel incorporado y sus vecinos que aun no pertenecen a la colonia. En suma, el número de mensajes intercambiados en este procedimiento tiene un costo en comunicaciones $C_{BFS} = O(|E| + |V| \log_k |V|)$. En cuanto a la complejidad en tiempo, considérese una colonia con n nodos, cuyo árbol interior tiene una altura $h \leq \log_k n$. Cada etapa de construcción toma, a lo mas, h unidades de tiempo y el total de etapas es h. Por tanto, el tiempo total que se requiere para formar una colonia y eliminar n nodos de la gráfica residual sera $O(\log_k^2 n)$. Así, el tiempo total invertido es $T_{BFS} = O(|V|)$.

Recién se termina la construcción de cierta colonia, el líder todavía en operación inicia la búsqueda del siguiente nodo que debe asumir este papel. El procedimiento puede tomar dos caminos diferentes: enviar un mensaje NEW_LEADER hacia un nodo en el nivel de rechazados de la colonia, o enviar un mensaje $RETREAT$ al nodo que lo designó como líder. En ambos casos el mensaje viaja solo a través de enlaces que pertenecen al árbol interior y, por tanto, la complejidad en mensajes es $C_M = O(|V|)$, mientras la complejidad en tiempo es $T_M = O(\log_k |V|)$.

Si se construye una gráfica $G' = (V', E')$ tal que $v \in V'$ representa una colonia resultante y $e \in E'$ es un enlace que une dos colonias, entonces puede decirse que el procedimiento de partición se desarrolla como un recorrido en profundidad (DFS) sobre G'. En consecuencia, el número de enlaces recorridos sobre esta gráfica inducida, es exactamente el número de colonias menos 1, que no puede ser mayor que $|V| - 1$ (suponiendo que cada nodo fuera una colonia diferente). En el

procedimiento DFS que se desarrolla, cada enlace es atravesado una vez en cada sentido, por tanto, la complejidad total del procedimiento de búsqueda será $C_{DFS} = O(|V|) \times C_M = O(|V|^2)$ y $T_{DFS} = O(|V|) \times T_M = O(|V| \log_k |V|)$, en cuanto a mensajes y tiempo, respectivamente. Finalmente, la selección de los enlaces preferidos implica una operación de convergencia de información a lo largo de cada árbol interior, con un costo $O(|V|)$ en mensajes y $O(\log_k |V|)$ en tiempo por cada colonia. Sumando sobre todas las colonias de la partición, se tiene una complejidad $C_{ELEC} = O(k|V|^2)$ y $T_{ELEC} = O(|V| \log_k |V|)$, en mensajes y tiempo respectivamente.

Así, la complejidad en comunicaciones es $C_{MST} + C_{BFS} + C_{DFS} + C_{ELEC}$ y en tiempo $T_{MST} + T_{BFS} + T_{DFS} + T_{ELEC}$. Por lo tanto, se concluye que la complejidad es

$$C_{TOTAL} = O(k|V|^2) \tag{3.2}$$

$$T_{TOTAL} = O(|V| \log_2 |V| / \log_2 k). \tag{3.3}$$

en comunicaciones y en tiempo, respectivamente.

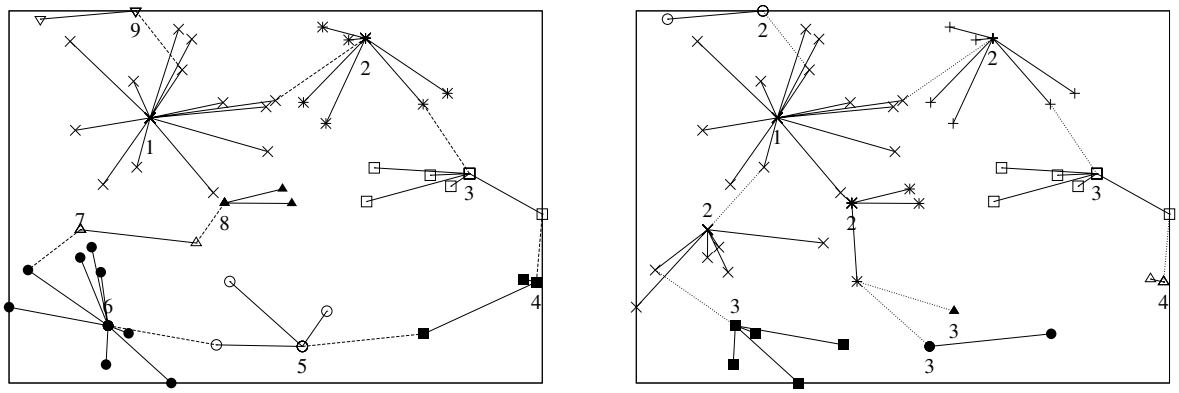
3.1.4. Modificación al Algoritmo de Particionamiento

En el algoritmo de partición la construcción de cada colonia no necesariamente debe ser secuencial. Con una pequeña modificación al algoritmo se puede conseguir que la partición se paralelice. Es decir, en lugar de seleccionar únicamente un candidato para crear una nueva colonia, se seleccionan varios candidatos para ser líderes y comienzan sus construcciones respectivas casi de manera simultánea.

Para ilustrar esta situación, en la red de la Figura 3.1A) se ejecutó el algoritmo con ambos criterios y el resultado se muestran en la Figuras 3.2A) y B). En las figuras, nótese que sobre cada colonia se colocaron números. Los números representan el tiempo en el que se construyeron, suponiendo que cada colonia se forma en una unidad de tiempo. A grandes rasgos, en la Figura 3.2 A) se observa que cada colonia se crea una tras otra y el particionamiento tomó nueve unidades de tiempo, es decir, se construyeron nueve colonias secuencialmente. En contraste, con el nuevo criterio de construcción mostrado en la Figura 3.2 B) se nota que varias colonias se construyen en el mismo instante; por ejemplo, alrededor de la primer colonia creada (la que tiene el número “1”) hay cuatro colonias que se crearon en el mismo instante: en el tiempo $t = 2$, también, hay cuatro colonias más que se crearon en $t = 3$ y así sucesivamente. Resumiendo, alrededor de una colonia se pueden construir varias colonias de manera simultánea. Por esta razón, la construcción de la partición con este criterio toma sólo cuatro unidades de tiempo.

Puede observarse que las colonias construidas con ambos algoritmos no son iguales, es decir, no tienen el mismo número de elementos o la misma topología. Esto se debe al tipo de criterio de partición empleado para la construcción. Sin embargo, la construcción individual de cada colonia se construye de la misma manera en ambos casos: mediante una búsqueda en amplitud.

Con referencia a la gráfica G' introducida con anterioridad puede decirse que el procedimiento original se desarrolló como un recorrido en profundidad (DFS) sobre G', mientras que el nuevo procedimiento se desarrolla como un recorrido en amplitud (BFS).



A) B)

Figura 3.2: A) Creación de las particiones en forma secuencial. B) Creación de las particiones en forma paralela.

El cambio del criterio DFS al BFS se consigue únicamente modificando uno de los mensajes empleados en la construcción de la colonia, este mensaje, llamado *CANDIDATO*, tiene tres funciones principales: seleccionar el candidato que será nuevo líder; realizar el retroceso (backtracking) en la gráfica particionada para continuar la búsqueda de los nuevos líderes; finalmente, disparar el proceso de selección de los enlaces preferidos.

En el Algoritmo 3.10 se muestra la estructura del mensaje *CANDIDATO* originalmente empleado en el algoritmo de Awerbuch. Cuando una colonia detiene su crecimiento los nodos *hoja* envían a su *padre* el candidato a nuevo líder *c* mediante el mensaje *CANDIDATO*. Cada nodo padre selecciona un único candidato (el de menor identificador) de todos los recibidos y, posteriormente, lo transmite a su padre. Cuando la información converge en el líder, éste selecciona el candidato final para comenzar a construir la nueva colonia.

Por otro lado, si no hay nodos disponibles para seleccionar un candidato, es decir, $c = NULO$. Entonces se *dispara* el procedimiento de elección del enlace preferido y se realiza el retroceso, el cual, cambia la acción a la colonia creada anteriormente para comenzar a buscar nuevos líderes en esta colonia.

Para que la creación de las colonias se realice en paralelo es necesario realizar ciertas modificaciones al mensaje *CANDIDATO*. Esto es, cuando los nodos hojas seleccionan un candidato, lo transmiten a su padre. El nodo padre, por su parte, agrega todos los candidatos recibidos en la lista $CANDIDATOS\{\}$, que posteriormente es transmitida a su padre correspondiente. Cuando la lista converge en el líder, cada nodo dentro de la lista es seleccionado como líder de colonia. Esta metodología es equivalente a una búsqueda en amplitud, ya que se seleccionan todos los candidatos posibles para continuar con el particionamiento. Debido a esta nueva estructura del mensaje *CANDIDATO*, la operación del retroceso ya no se utiliza, tal y como se muestra en el Algoritmo 3.11.

Algoritmo 3.10: Mensaje *CANDIDATO* del algoritmo de partición de Awerbuch

```
/* Una vez que los nodos hoja han seleccionado su candidato, envían el
   mensaje CANDIDATO{c} a su padre. Donde c es el candidato a ser
   nuevo líder                                                        */
para el mensaje CANDIDATO{c} de q haz
    Candidato(i) ← mín{Candidato(i), c};
    // Los nodos padre seleccionan un sólo candidato (el de menor
    identificador de todos los candidatos recibidos)
    si He recibido los candidatos de todos mis hijos entonces
        si no soy líder entonces
            enviar CANDIDATO{Candidato(i)} a Padre
        fin
        otro // Si soy líder
            si Candidato(i) = c ≠ NULO entonces // Si aún hay candidatos
            disponibles
                seleccionar a c como nuevo líder;
            fin
            otro
                iniciar elección de los enlaces preferidos;
                iniciar retroceso;
                // retroceder la acción a la colonia anterior para seguir
                buscando nuevos líderes (Backtrack)
            fin
        fin
    fin
fin
```

Algoritmo 3.11: Mensaje *CANDIDATO* del algoritmo de partición modificado

```
/* Una vez que los nodos hoja han seleccionado su candidato, envían el
   mensaje CANDIDATO{c} a su padre. Donde c es el candidato a ser
   nuevo líder                                                        */
para el mensaje CANDIDATO{c} de q haz
    CANDIDATOS{i} ← c;
    // Los nodos padre agregan a la lista CANDIDATOS{} todos los
    candidatos enviados por sus hijos (se eliminan duplicados)
    si He recibido los candidatos de todos mis hijos entonces
        si no soy líder entonces // Enviar la lista de candidatos a mi Padre
            enviar CANDIDATO{CANDIDATOS{i}} a Padre;
        fin
        otro // Si soy líder
            seleccionar a cada j ∈ CANDIDATOS{j} como líder de colonia;
            iniciar elección de los enlaces preferidos;
        fin
    fin
fin
```

3.1.5. Complejidad del Algoritmo de Particionamiento Modificado

La partición de la red mediante el algoritmo de Awerbuch emplea tres procedimientos básicos: *Creación_de_colonia*, en el cual se construye el árbol BFS que conforma la colonia; *Búsqueda_del_líder*, es el procedimiento encargado de buscar el nuevo líder de colonia; finalmente, la *Elección_del_enlace_preferido*, en el cual se seleccionan aquellos enlaces que comunican a dos colonias contiguas. Estos procesos no se modifican en absoluto, salvo la manera en que cada uno de ellos se desarrolla en cada algoritmo.

La modificación al algoritmo de particionamiento implica que los procedimientos involucrados en la creación de la colonia se ejecuten una sola vez, por ejemplo, en Awerbuch, el procedimiento *Búsqueda_del_líder* se ejecuta en cada una de las colonias cada vez que la actividad realiza un rastreo en retroceso (backtracking) para buscar nuevos líderes en la gráfica residual. También, el procedimiento *Elección_del_enlace_preferido* se ejecuta varias veces en una sola colonia cada vez que una nueva colonia contigua a ella se crea. Así, *paralelizando* la creación de cada colonia, el procedimiento *Búsqueda_del_líder* se ejecuta una sola vez en cada colonia debido a que se eliminan los retrocesos concernientes a la búsqueda DFS. Por otro lado la *Elección_del_enlace_ preferido* también se ejecuta una sola vez en cada colonia, ya que todas las colonias que se pueden crear alrededor de otra se construyen de manera simultánea.

En conclusión, las complejidades del algoritmo de Awerbuch son válidas para el algoritmo modificado ya que son las mismas, por tanto, la complejidad es:

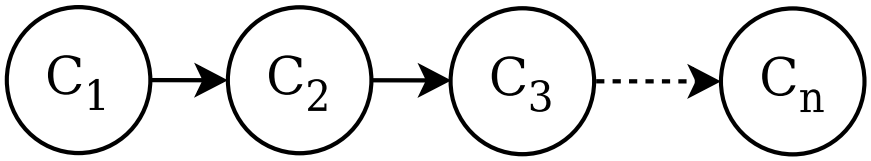


Figura 3.3: Resultado del particionamiento en el peor caso.

$$
\begin{aligned}
C_{MST} &= O(|V|^2), \quad T_{MST} = O(|V| \log_2 |V|) \\
C_{BFS} &= O(|E| + |V| \log_k |V|), \quad T_{BFS} = O(|V| \log_k |V|) \\
C_{DFS} &= O(|V|^2), \quad T_{DFS} = O(|V| \log_k |V|) \\
C_{ELEC} &= O(k|V|^2), \quad T_{ELEC} = O(|V| \log_k |V|)
\end{aligned}
$$

Lo que implica que el costo en comunicaciones y en tiempo son

$$C_{TOTAL} = O(k|V|^2) \tag{3.4}$$

$$T_{TOTAL} = O(|V| \log_2 |V| / \log_2 k) \tag{3.5}$$

Cabe mencionar la complejidad del algoritmo modificado en el peor caso es exactamente la misma que Awerbuch. Por ejemplo, cuando la red se particiona de tal manera que cada colonia se construye una tras otra implicaría que las colonias no se creen de manera simultánea sino secuencial, provocando que cada una de las subrutinas mencionadas anteriormente se ejecuten también de secuencialmente (Figura 3.3). En este caso, el comportamiento del algoritmo modificado es muy similar al algoritmo de Awerbuch.

Resumiendo, el algoritmo modificado concluye, en promedio, en menor tiempo que Awerbuch. Además, algunas subrutinas sólo se ejecutan una sola vez durante el algoritmo, disminuyendo con ello el costo en comunicaciones. Sin embargo, Awerbuch tiene *terminación*, es decir, que el nodo que inició el algoritmo de partición tiene conocimiento de cuando este termina. En cambio, el algoritmo modificado no tiene terminación, esta es una razón más por la que siempre finaliza en menor tiempo.

3.2. Algoritmo Bellman-Ford

Consideremos un conjunto V de ciudades, numeradas de una manera arbitraria, donde $i, j \in V$. Se asume que existe una trayectoria entre cada par de ciudades. La distancia de la ciudad i a j no necesariamente es proporcional a la distancia de j a i. Dada la matriz $D = [d_{ij}]$, no necesariamente simétrica, donde d_{ij} es la distancia de i a j, se desea encontrar la trayectoria más corta entre éstas. Debido a que sólo un número finito de trayectorias está disponible, el problema se reduce a seleccionar la más pequeña de un número finito. En 1957, Bellman [5], propone una técnica capaz

de reducir el tiempo requerido para encontrar las trayectorias más cortas mediante programación dinámica. Este problema es un problema combinatorio que requiere determinar la ruta óptima de un punto a otro.

El método básico es por aproximaciones sucesivas. Seleccionamos una secuencia inicial

$$f_{ij}^{(0)} = d_{ij}, \quad i, j \in V \tag{3.6}$$

con $d_{ii} = 0$ y $d_{ij} = \infty$ si no hay un enlace directo entre i y j.

Ahora bien, procediendo iterativamente, tenemos que

$$f_{ij}^{(l+1)} = \min_{k \neq j} \left[f_{ij}^{(l)}, f_{ik}^{(l)} + d_{kj} \right], \tag{3.7}$$

El algoritmo Bellman-Ford [11] puede enunciarse de la siguiente forma: encontrar los caminos más cortos desde un nodo origen único hacia todos los nodos de destino con la condición de que éstos se encuentren a lo más a un enlace de distancia; a continuación, encontrar los caminos más cortos con la condición de que contengan dos enlaces como máximo, y así sucesivamente. Para esta tarea, la ecuación (3.7) nos proporcionará las mejores trayectorias para ir de un punto a otro. Este algoritmo itera sobre el número de saltos, h, es decir, se busca el mejor camino, el de distancia más corta, con la restricción de llegar a los nodos de destino en un número de saltos h. En general, consiste fundamentalmente en calcular la distancia de un nodo (aquel para el cual aplicamos el algoritmo, al que llamamos nodo origen) hacia todos los demás nodos de la red, incrementando en alcance de la búsqueda en cada iteración. El algoritmo termina cuando se evalúan todas las rutas que parten del nodo origen y llegan a todos los destinos alcanzables [16].

En concreto, sea $G = (V, E)$ una gráfica con una matriz de adyacencias **D** asociada a $i, j \in V$. El vértice i, que ejecuta el algoritmo, comienza con la matriz de adyacencias de G, $D = [d_{ij}]$, donde cada d_{ij} es un estimado del costo de la trayectoria mínima desde el nodo i hacia todos los vértices $j \in V$. Además $d_{ii} = 0$ y $d_{ij} = \infty$, cuando no existe un enlace directo entre i y j. También sea $f_{ij}^{(l)}$ la distancia calculada, al terminar la iteración l, para viajar desde i hacia todo $j \in V$. Ahora, el nodo i emplea el método iterativo de Bellman (ecuación (3.7)) para optimizar el vector $f_{ij}^{(l+1)}$ desde i hacia todo $j \in V$. El algoritmo puede emplearse para estimar las mejores rutas entre cada par de nodos únicamente con el conocimiento de la matriz de adyacencias. Así, si un nodo cuenta con esta matriz relacionada con una red (o a una colonia), puede estimar las mejores rutas de manera centralizada, tal y como se muestra en el Algoritmo 3.12.

3.2.1. Complejidad del Algoritmo Bellman-Ford

De la ecuación (3.7), es claro que este esquema iterativo requiere a lo más $(|V| - 1)$ iteraciones para que la secuencia obtenga la solución final. Sin embargo, en cada iteración se revisa el costo para encontrar la trayectoria más corta hacia un único nodo desde cualquier otro de los $(V - 1)$ restantes. Entonces, se requieren $(|V| - 1)(|V| - 1)$ iteraciones. Finalmente, para que cada nodo estime las mejores rutas hacia cualquier nodo, se debe repetir el procedimiento $(|V| - 1)$ veces, dando como resultado $(|V| - 1)(|V| - 1)(|V| - 1)$. Así, el orden de la complejidad en tiempo del algoritmo Bellman-Ford es

Algoritmo 3.12: Algoritmo Bellman-Ford

Entrada: Matriz de adyacencias $\mathbf{D} = [d_{ij}]$

Resultado: Se completa la matriz $M = [min_{ij}]$ a partir de la matriz de adyacencias con el camino de costo mínimo y el costo asociado a cada nodo.

Datos:

d_{ij} Costo del enlace de i a j.

min_{ij} Costo del mínimo de la ruta de i a j

e_{ij} Costo actual estimado de la ruta de i a j

tam Tamaño de la matriz **D**

```
Inicialización;
tam=D.tamaño;
mientras No haya más cambios en M haz
    para i=1 hasta tam haz
        para j=1 hasta tam haz
            min=∞;
            para k=1 hasta tam haz
                e_ij=d_ki+d_jk;
                si e_ij ≤min entonces
                    min=e_ij;
                fin
            fin
            min_ij=min;
        fin
    fin
fin
```

$$T_{BF} = O(|V|^3) \tag{3.8}$$

3.3. Escalamiento Multi-Dimensional

Una definición general de la técnica de análisis de datos llamada *Escalamiento Multi-Dimensional* (MDS) es la búsqueda de un espacio dimensional, usualmente *Euclidiano*, en el cual los puntos en el espacio representan los objetos de tal forma que las distancias estimadas entre los puntos del espacio $\{d_{rs}\}$, concuerdan lo más posible con las distancias reales de los objetos representados $\{\delta_{rs}\}$ [8]. Estas distancias son llamadas comúnmente *disimilaridades*.

Tabla 3.1: Distancias reales (disimilaridades) entre diez ciudades europeas.

-	Lond	Estoc	Lisb	Madr	París	Amst	Berln	Praga	Roma	Dubln
Lond	0	569	667	530	141	140	357	396	570	190
Estoc	569	0	1212	1043	617	446	325	423	787	648
Lisb	667	1212	0	201	596	768	923	882	714	714
Madr	530	1043	201	0	431	608	740	690	516	622
París	141	617	596	431	0	177	340	337	436	320
Amst	140	446	768	608	177	0	218	272	519	302
Berln	357	325	923	740	340	218	0	114	472	514
Praga	396	423	882	690	337	272	114	0	364	573
Roma	569	787	714	516	436	519	472	364	0	755
Dubln	190	648	714	622	320	302	514	573	755	0

Las técnicas usadas para la búsqueda del espacio y la configuración asociada de puntos forman un Escalamiento Multi-Dimensional métrico o no métrico.

Supongamos que tenemos n objetos con disimilaridades $\{\delta_{rs}\}$. El MDS intenta encontrar un conjunto de puntos en un espacio donde cada punto representa uno de los objetos y las distancias entre los puntos $\{d_{rs}\}$ son tales que $d_{rs} \approx f(\delta_{rs})$, donde f es una función paramétrica monótona. La función f puede ser una función identidad o una función que intenta transformar las disimilaridades en una forma de distancias.

Para ilustrar el MDS consideremos la Tabla 3.1, en la cual, se muestran las distancias medidas (en millas) entre diez ciudades europeas. El MDS trata de revertir este proceso. Esto es, basados únicamente en las disimilaridades, trataremos de encontrar una configuración de diez puntos en el plano tal que las distancias entre esos diez puntos correspondan a las distancias entre las diez ciudades del mapa original, aún cuando el mapa reconstruido tenga una orientación arbitraria.

La Figura 3.4 muestra la configuración de puntos producidos por el MDS. Nótese que el mapa reconstruido no cuenta con la orientación convencional, sin embargo, esto puede ajustarse fácilmente. En primer lugar, supongamos que la ciudad de París es un *pivote*. Ahora, debemos rotar todo el mapa en sentido de las manecillas del reloj alrededor de París. La rotación se debe realizar de tal manera que Estocolmo sea la ciudad que se encuentre más hacia el norte y hacia el este.

Las transformaciones que pueden aplicarse a una configuración MDS son rotación, traslación y reflexión, las cuales, son transformaciones lineales que preservan las distancias[3]. Sin embargo, algunas transformaciones que no preservan distancias, como los estiramientos, son válidos también ya que, aunque las distancias se ven alteradas, la proporción de las distancias entre cada par de puntos se mantiene.

Matemáticamente, los objetos componen un conjunto O. La disimilaridad, definida en $O \times O$, entre los objetos r y s es δ_{rs} $(r, s \in O)$. Sea ϕ una transformación arbitraria de O a E, donde E es un espacio Euclidiano en la cual los puntos van a representar a los objetos. Entonces $\phi(r) = x_r$ $(r \in O, x_r \in E)$ y $X = \{x_r : r \in O\}$ es el conjunto imagen. La distancia entre los puntos x_r, x_s en X son dadas por δ_{rs}. El objetivo es encontrar una transformación ϕ, para la cual δ_{rs} sea aproximadamente

[3]Las transformaciones lineales que preservan la longitud son llamadas isometrías

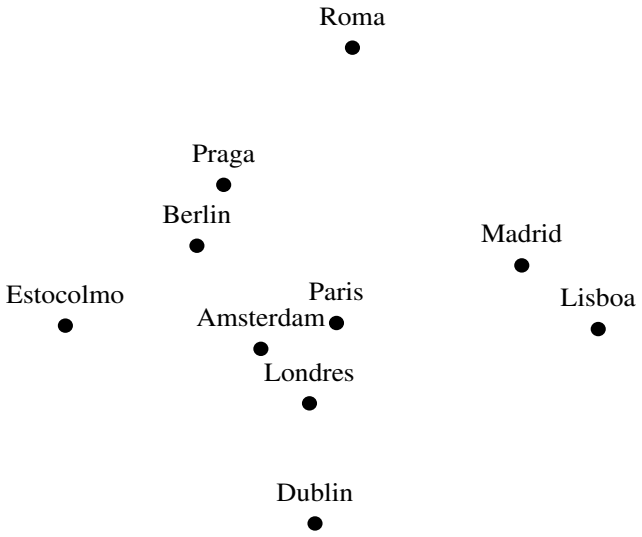


Figura 3.4: Mapa de las diez ciudades europeas reconstruido con la distancia entre cada una de ellas.

igual a $f(\delta_{rs})$ para todo $r, s \in O$.

3.3.1. Calculando Distancias Empleando Álgebra de Matrices

Un concepto importante en el MDS es la distancia entre dos puntos. Sea $\mathbf{X} \in \mathbf{M}_{n\times m}$ la matriz de las coordenadas de n puntos. Cada fila i de $\mathbf{X}$ da las coordenadas del punto i en m dimensiones. Empleando álgebra de matrices, es posible encontrar una expresión compacta para calcular el cuadrado de las distancias Euclidianas entre todos los puntos. El cuadrado de la distancia Euclidiana está definido por:

$$d_{ij}^2(\mathbf{X}) = d_{ij}^2 = \sum_{a=1}^{m}(x_{ia} - x_{ja})^2 = \sum_{a=1}^{m}(x_{ia}^2 + x_{ja}^2 - 2x_{ia}x_{ja}) \tag{3.9}$$

Supóngase que $\mathbf{X}$ contiene las coordenadas de tres puntos en dos dimensiones. Entonces, la matriz del cuadrado de las distancias, denotado por $\mathbf{D}^{(2)}(\mathbf{X})$, es

$$\begin{aligned}\mathbf{D}^{(2)}(\mathbf{X}) &= \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 \\ d_{12}^2 & 0 & d_{23}^2 \\ d_{13}^2 & d_{23}^2 & 0 \end{bmatrix} = \sum_{a=1}^{m} \begin{bmatrix} x_{1a}^2 & x_{1a}^2 & x_{1a}^2 \\ x_{2a}^2 & x_{2a}^2 & x_{2a}^2 \\ x_{3a}^2 & x_{3a}^2 & x_{3a}^2 \end{bmatrix} + \sum_{a=1}^{m} \begin{bmatrix} x_{1a}^2 & x_{1a}^2 & x_{1a}^2 \\ x_{2a}^2 & x_{2a}^2 & x_{2a}^2 \\ x_{3a}^2 & x_{3a}^2 & x_{3a}^2 \end{bmatrix} - \\ & 2\sum_{a=1}^{m} \begin{bmatrix} x_{1a}x_{1a} & x_{1a}x_{2a} & x_{1a}x_{3a} \\ x_{2a}x_{1a} & x_{2a}x_{2a} & x_{2a}x_{3a} \\ x_{3a}x_{1a} & x_{3a}x_{2a} & x_{3a}x_{3a} \end{bmatrix} \\ &= \mathbf{c1}^T + \mathbf{1c}^T - 2\sum_{a=1}^{m} x_a x_a^T = \mathbf{c1}^T + \mathbf{1c}^T - 2\mathbf{XX}^T \end{aligned} \tag{3.10}$$

donde x_a es la columna a de la matriz $\mathbf{X}$ y $\mathbf{c}$ es un vector cuyos elementos son $\sum_{a=1}^{m} x_{ia}^2$, los elementos de la diagonal de $\mathbf{XX}^T$. La matriz $\mathbf{B} = \mathbf{XX}^T$, es llamada *matriz producto escalar* o también *matriz producto interno*.

Supóngase que

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 0 \end{bmatrix} \tag{3.11}$$

es una matriz de coordenadas, de tres puntos en 2 dimensiones. Las distancias pueden calcularse empleando la ecuación (3.10). Entonces, el primer paso es calcular la matriz producto escalar $\mathbf{B} = \mathbf{XX}^T$.

$$\mathbf{XX}^T = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 5 & 2 \\ 5 & 10 & 6 \\ 2 & 6 & 4 \end{bmatrix} = \mathbf{B} \tag{3.12}$$

el segundo paso es encontrar $\mathbf{c}$. Puede observarse que los elementos de la diagonal de la matriz $\mathbf{XX}^T$ son $\sum_{a=1}^{m} x_{ia}^2$, los cuales son los elementos de $\mathbf{c}$. Así, $\mathbf{c} = (5, 10, 4)$. Sustituyendo en la ecuación (3.10) se tiene

$$\begin{aligned}\mathbf{D}^{(2)}(\mathbf{X}) &= \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 \\ d_{12}^2 & 0 & d_{23}^2 \\ d_{13}^2 & d_{23}^2 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 5 & 5 \\ 10 & 10 & 10 \\ 4 & 4 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 10 & 4 \\ 5 & 10 & 4 \\ 5 & 10 & 4 \end{bmatrix} \\ & -2\begin{bmatrix} 5 & 5 & 2 \\ 5 & 10 & 6 \\ 2 & 6 & 4 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 5 \\ 5 & 0 & 2 \\ 5 & 2 & 0 \end{bmatrix} \end{aligned} \tag{3.13}$$

Tomando la raíz cuadrada de todos los elementos da la matriz de distancias.

$$\mathbf{D(X)} = \begin{bmatrix} 0 & \sqrt{5} & \sqrt{5} \\ \sqrt{5} & 0 & \sqrt{2} \\ \sqrt{5} & \sqrt{2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2.236 & 2.236 \\ 2.236 & 0 & 1.414 \\ 2.236 & 1.414 & 0 \end{bmatrix} \tag{3.14}$$

En la sección 3.3.5 se muestra cómo se puede resolver el problema inverso, es decir, cómo encontrar las coordenadas que corresponden a una matriz **X** a partir de una matriz producto escalar **B** dada.

3.3.2. Descomposición Espectral

Cada matriz **A** de $n \times n$ puede descomponerse en el producto de varias matrices mas sencillas. Una de estas descomposiciones es la *eigendescomposición*, la cual puede realizarse para la mayoría de las matrices, sin embargo, siempre funciona para las matrices simétricas. Formalmente,

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T \tag{3.15}$$

con **Q** ortonormal y Λ es una matriz diagonal. Las columnas de Q representan los eigenvectores de A y la diagonal de Λ contiene los eigenvalores. La ecuación (3.15) puede escribirse como

$$\mathbf{A}\mathbf{q}_i = \lambda_i \mathbf{q}_i \tag{3.16}$$

con $\mathbf{q}_i \neq 0$ (i=1,...,n).

Los vectores q_i son llamados *eigenvectores* de **A** y las λ_i son llamadas *eigenvalores* de **A**. Por ejemplo, para la matriz A

$$\mathbf{A} = \begin{bmatrix} 23 & 36 \\ 36 & 2 \end{bmatrix}$$

tenemos

$$\mathbf{Q} = \begin{bmatrix} 0.8 & -0.6 \\ 0.6 & 0.8 \end{bmatrix}$$

y

$$\Lambda = \begin{bmatrix} 50 & 0 \\ 0 & -25 \end{bmatrix}$$

Considérese nuevamente la ecuación (3.15). Puede pensarse que el producto $Q\Lambda Q^T$ es un producto de dos vectores: el vector fila está compuesto por el producto $Q\Lambda$, así, el vector columna se compone de los vectores de Q^T

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \lambda_1 q_1 & \lambda_2 q_2 & \dots & \lambda_n q_n \end{bmatrix} \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_n^T \end{bmatrix} \\ &= \lambda_1 q_1 q_1^T + \lambda_2 q_2 q_2^T + \cdots + \lambda_n q_n q_n^T \end{aligned} \tag{3.17}$$

La ecuación (3.17) dice que **A** puede descomponerse como una suma de matrices. Para ilustrar esto, consideremos nuevamente la matriz **A** mostrada anteriormente. La descomposición es

$$\begin{aligned}\mathbf{A} &= 50\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}\begin{bmatrix} 0.8 & 0.6 \end{bmatrix} - 25\begin{bmatrix} -0.6 \\ 0.8 \end{bmatrix}\begin{bmatrix} -0.6 & 0.8 \end{bmatrix} \\ &= \begin{bmatrix} 32 & 24 \\ 24 & 18 \end{bmatrix} - \begin{bmatrix} 9 & -12 \\ -12 & 16 \end{bmatrix} = \begin{bmatrix} 23 & 36 \\ 36 & 2 \end{bmatrix}\end{aligned}$$

3.3.3. Método Iterativo para Realizar Eigendescomposiciones

La manera habitual de encontrar los eigenvalores y eigenvectores de una matriz **A** es encontrar las raíces λ del determinante

$$|\mathbf{A} - \lambda\mathbf{I}|$$

A partir de esto encontramos los eigenvectores que dan solución a la ecuación

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{q} = 0$$

Esto es, sin embargo, ineficaz cuando **A** es muy grande. Inclusive, encontrar el determinante de una matriz de $n \times n$ es un arduo trabajo cuando n es grande. Otra complicación más es que se tiene que resolver el polinomio resultante de grado n para λ. Por esto, se mostrará un método iterativo para realizar la eigendescomposición de una matriz, empleando, un método numérico para estimar el valor aproximado de los eigenvalores y eigenvectores de una matriz de dimensiones $n \times n$. Este método es llamado *método de las potencias* (power method) [13] y se emplea en distintos paquetes computacionales.

El método de las potencias se utiliza cuando los eigenvalores son diferentes y pueden ordenarse como, $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots |\lambda_n|$. Se asume que esto se cumple cuando se trabaja con matrices producto escalar (véase la sección 3.3.1).

Considérese la matriz producto escalar **B** de la ecuación (3.12). Se define arbitrariamente un vector de inicio $\mathbf{q}^{[0]} \neq \mathbf{0}$ e iteramos el sistema

$$\mathbf{q}^{[t+1]} = \frac{\mathbf{B}\mathbf{q}^{[t]}}{(\mathbf{q}^{[t]T}\mathbf{B}^T\mathbf{B}\mathbf{q}^{[t]})^{1/2}} \tag{3.18}$$

la cantidad de veces necesarias hasta que $\mathbf{q}^{[t+1]}$ permanece esencialmente invariante sobre las otras iteraciones t. Después de la convergencia, $\mathbf{q}$ es igual al primer eigenvector y $\mathbf{q}^T\mathbf{B}\mathbf{q} = \lambda_1$. El primer y segundo eigenvalor se muestran en la Tabla 3.2.

Iniciando con $\mathbf{q}^{[0]} = (3^{-1/2}, 3^{-1/2}, 3^{-1/2})$, $\mathbf{B}\mathbf{q}^{[0]}$ =(6.928, 12.124, 6.928) y $\mathbf{q}^{[0]T}\mathbf{B}^T\mathbf{B}\mathbf{q}^{[0]} = 243$. Así, $\mathbf{q}^1$ =(6.928, 12.124, 6.928)/$243^{1/2}$ =(0.444, 0.778, 0.444). La segunda iteración se obtiene $\mathbf{B}\mathbf{q}^{[1]}/(\mathbf{q}^{[1]T}\mathbf{B}^T\mathbf{B}\mathbf{q}^{[1]})^{1/2}$ =(0.431, 0.781, 0.452). Después de cuatro iteraciones $\mathbf{q}^{[4]}$ =(0.429, 0.781, 0.454), el cual corresponde al primer eigenvector. De igual manera, el primer eigenvalor λ_1 se obtiene de $(\mathbf{q}^{[4]T}\mathbf{B}^T\mathbf{B}\mathbf{q}^{[4]})^{1/2}$ =16.227.

Ahora bien, para encontrar el segundo eigenvector, en la ecuación (3.17), la eigendescomposición de una matriz de 3×3 es

$$\mathbf{B} = \lambda_1\mathbf{q}_1\mathbf{q}_1^T + \lambda_2\mathbf{q}_2\mathbf{q}_2^T + \lambda_3\mathbf{q}_3\mathbf{q}_3^T$$

Tabla 3.2: Eigenvalores y eigenvectores obtenidos mediante el método de las potencias

B			$\mathbf{q}^{[0]}$	$\mathbf{q}^{[1]}$	$\mathbf{q}^{[2]}$	$\mathbf{q}^{[3]}$	$\mathbf{q}^{[4]}$	$\mathbf{q}_1$
5	5	2	$3^{-1/2}$	0.444	0.431	0.429	0.429	0.429
5	10	6	$3^{-1/2}$	0.778	0.781	0.781	0.781	0.781
2	6	4	$3^{-1/2}$	0.444	0.452	0.453	0.454	0.454
		$\lambda^{[t]}$	15.588	16.224	16.227	16.227	16.227	16.227

$\mathbf{B} - \lambda_1\mathbf{q}_1\mathbf{q}_1^T$			$\mathbf{q}^{[0]}$	$\mathbf{q}^{[1]}$	$\mathbf{q}^{[2]}$	$\mathbf{q}_2$
2.016	-0.437	-1.156	$3^{-1/2}$	0.853	0.853	0.853
-0.437	0.95	0.251	$3^{-1/2}$	-0.185	-0.185	-0.185
-1.156	0.251	0.663	$3^{-1/2}$	-0.489	-0.489	-0.489
		$\lambda^{[t]}$	0.286	2.773	2.773	2.773

En este punto, ya se conoce el primer eigenvalor λ_1 y el primer eigenvector $\mathbf{q}_1$. Entonces

$$\mathbf{B} - \lambda_1\mathbf{q}_1\mathbf{q}_1^T = \lambda_2\mathbf{q}_2\mathbf{q}_2^T + \lambda_3\mathbf{q}_3\mathbf{q}_3^T$$

Para calcular el segundo eigenvalor y eigenvector, se aplica nuevamente el procedimiento a la matriz $\mathbf{B} - \lambda_1\mathbf{q}_1\mathbf{q}_1^T$. Esto se muestra en la segunda parte de la Tabla 3.2. El eigenvector $\mathbf{q}_2$ es (0.853, -0.185, -0.489) y λ_1 es 2.773. Para encontrar el tercer eigenvalor, es necesario repetir el procedimiento a la matriz $\mathbf{B} - \lambda_1\mathbf{q}_1\mathbf{q}_1^T - \lambda_2\mathbf{q}_2\mathbf{q}_2^T$, el cual, en este ejemplo particular es cero. De esta manera,

$$\lambda_1\mathbf{q}_1\mathbf{q}_1^T = \begin{bmatrix} 2.984 & 5.437 & 3.156 \\ 5.437 & 9.905 & 5.749 \\ 3.156 & 5.749 & 3.337 \end{bmatrix}, \ \lambda_2\mathbf{q}_2\mathbf{q}_2^T = \begin{bmatrix} 2.016 & -0.437 & -1.156 \\ -0.437 & 0.095 & 0.251 \\ -1.156 & 0.251 & 0.663 \end{bmatrix}$$

Así

$$\lambda_1\mathbf{q}_1\mathbf{q}_1^T + \lambda_2\mathbf{q}_2\mathbf{q}_2^T = \begin{bmatrix} 5 & 5 & 2 \\ 5 & 10 & 6 \\ 2 & 6 & 4 \end{bmatrix} = \mathbf{B}$$

3.3.4. Complejidad del Método de las Potencias

En resumen, el método de las potencias produce una secuencia de vectores $\mathbf{q}^{[t]}$ como sigue

$$\mathbf{q}^{[t+1]} = \frac{\mathbf{B}\mathbf{q}^{[t]}}{(\mathbf{q}^{[t]T}\mathbf{B}^T\mathbf{B}\mathbf{q}^{[t]})^{1/2}}$$

Este método converge si el primer eigenvalor obtenido λ_1 es dominante, es decir es el mayor de todos, lo cual se asume que es verdad cuando se emplea una matriz producto escalar como la que se muestra en la ecuación (3.12). Nótese que el método de las potencias precisa únicamente de

multiplicaciones sencillas de matrices. El producto $\mathbf{Bq}^{[t]}$ requiere de n^2 pasos puesto que $\mathbf{B}$ es una matriz de $n \times n$ y $\mathbf{q}^{[t]}$ es una matriz $n \times 1$. De igual manera, para la multiplicación $\mathbf{q}^{[t]T}\mathbf{B}^T\mathbf{Bq}^{[t]}$ se tiene que el producto $\mathbf{q}^{[t]T}\mathbf{B}^T$ requiere de n^2 pasos, ya que es el producto de una matriz de $1 \times n$ por una matriz de $n \times n$, esto produce una matriz de dimensión $1 \times n$. Posteriormente, este resultado se multiplica por $\mathbf{B}$ cuya dimensión es $n \times n$. Este producto requiere, de igual manera, n^2 pasos. Finalmente, la multiplicación de este último resultado por $\mathbf{q}$ requiere únicamente n pasos, debido a que es el producto de una matriz de $1 \times n$ por una de $n \times 1$.

Sumando las complejidades de cada uno de los productos se tiene que el costo del sistema es $3n^2 + n$. Sin embargo, este sistema itera i veces para obtener una única secuencia de eigenvectores y su correspondiente eigenvalor. Esto implica que para obtener los n eigenvectores y los n eigenvalores, entonces el sistema debe multiplicarse por $i \times n$, dando como resultado una complejidad en tiempo de

$$T_{PM} = O(i \times n \times (3n^2 + n)) = O(i \times n^3) \tag{3.19}$$

Nótese que el número de pasos necesarios para realizar el producto $\mathbf{q}^{[t]T}\mathbf{B}^T\mathbf{Bq}^{[t]}$ depende, de la manera en que las matrices se asocian, esto es, si la multiplicación se realiza en el orden aquí explicado, entonces la complejidad es $2n^2 + n$. Sin embargo, si el producto se realiza asociando las matrices de la siguiente manera: $\mathbf{q}^{[t]T}(\mathbf{B}^T\mathbf{B})\mathbf{q}^{[t]}$, entonces la complejidad resultante del sistema es $n^3 + n^2 + n$ puesto que el producto $\mathbf{B}^T\mathbf{B}$ requiere de n^3 pasos. Así, la complejidad final, agregando el sobreprecio de $i \times n$ es $O(i \times n^4)$.

3.3.5. Configuraciones que Representan Productos Escalares

Retomando el problema de encontrar una configuración de puntos que representan una matriz producto escalar dada, esto es, resolver la ecuación

$$\mathbf{B} = \mathbf{XX}^T \tag{3.20}$$

donde $\mathbf{X}$ es una matriz de coordenadas de $n \times m$, es decir, n puntos en m dimensiones. Sea

$$\mathbf{X} = \begin{bmatrix} 1 & 2 \\ 3 & 1 \\ 2 & 0 \end{bmatrix} \tag{3.21}$$

y

$$\mathbf{B} = \mathbf{XX}^T = \begin{bmatrix} 5 & 5 & 2 \\ 5 & 10 & 6 \\ 2 & 6 & 10 \end{bmatrix} \tag{3.22}$$

Supóngase que se realiza la descomposición espectral de $\mathbf{B} = \mathbf{Q}\Lambda\mathbf{Q}^T$. El producto escalar de matrices es simétrico y tiene eigenvalores no negativos. Por esto, $\mathbf{B}$ se puede reescribir como $\mathbf{B} = (\mathbf{Q}\Lambda^{1/2})(\mathbf{Q}\Lambda^{1/2})^T = \mathbf{UU}^T$ donde $\Lambda^{1/2}$ es una matriz diagonal con elementos $\lambda_i^{1/2}$. De esta manera, $\mathbf{U} = \mathbf{Q}\Lambda^{1/2}$ da las coordenadas para reconstruir B. Las coordenadas son

$$\begin{aligned}\mathbf{U} &= \mathbf{Q}\Lambda^{1/2} \\ &= \begin{bmatrix} 0.43 & 0.85 \\ 0.78 & -0.19 \\ 0.45 & -0.49 \end{bmatrix}\begin{bmatrix} 4.03 & 0 \\ 0 & 1.67 \end{bmatrix} = \begin{bmatrix} 1.73 & 1.42 \\ 3.15 & -0.31 \\ 1.83 & -0.81 \end{bmatrix}\end{aligned} \tag{3.23}$$

Las coordenadas de **U** difieren de las de **X** en (3.21), esto significa que ambas están expresadas en dos sistemas de coordenadas relativos. Sin embargo, los sistemas pueden rotarse uno sobre otro. Para el problema de encontrar la configuración de vectores de un producto escalar, es irrelevante cómo los ejes coordenados se rotan, en cambio, lo que importa es la configuración de todos los puntos.

3.3.6. MDS Clásico

En el ejemplo de las diez ciudades Europeas (Sección 3.3), suponiendo que el continente Europeo se representa en un espacio Euclidiano de dos dimensiones, ¿pueden encontrarse las posiciones originales de las ciudades? La respuesta es si, pero sólo las posiciones relativas de una con otra. Debido a que cualquier solución puede trasladarse, rotarse, reflejarse y escalarse se puede obtener una solución de igual validez.

La idea principal del MDS clásico [13] es asumir que las disimilaridades son distancias y entonces, encontrar las coordenadas que los expliquen. La ecuación (3.10) es una expresión sencilla para obtener el cuadrado de las distancias Euclidianas (**D(X)**) a partir de una matriz de coordenadas **X**. En la sección 3.3.5 se discute el procedimiento inverso, esto es: obtener la matriz de coordenadas a partir del producto escalar $\mathbf{B} = \mathbf{X}\mathbf{X}^T$. El MDS clásico emplea los mismos procedimientos pero trabaja sobre la matriz de disimilaridades Δ^2 en vez de $\mathbf{D(X)}^2$.

Encontrando Coordenadas con MDS Clásico

En la ecuación (3.10) las distancias al cuadrado de X se calculan con

$$\mathbf{D(X)}^2 = \mathbf{c1}^T + \mathbf{1c}^T - 2\mathbf{X}\mathbf{X}^T = \mathbf{c1}^T + \mathbf{1c}^T - 2\mathbf{B} \tag{3.24}$$

donde **c** es un vector con los elementos de la diagonal de $\mathbf{X}\mathbf{X}^T$. Multiplicando los lados izquierdos y derechos por la matriz centradora $\mathbf{H} = \mathbf{I} - \mathbf{n}^{-1}\mathbf{1}\mathbf{1}^T$ y por el factor $-\frac{1}{2}$ tenemos

$$\begin{aligned}-\frac{1}{2}\mathbf{H}\mathbf{D(X)}^2\mathbf{H} &= -\frac{1}{2}\mathbf{H}(\mathbf{c1}^T + \mathbf{1c}^T - 2\mathbf{X}\mathbf{X}^T)\mathbf{H} \\ &= -\frac{1}{2}\mathbf{H}\mathbf{c1}^T\mathbf{H} - \frac{1}{2}\mathbf{H}\mathbf{1c}^T\mathbf{H} + \frac{1}{2}\mathbf{H}(2\mathbf{B})\mathbf{H} \\ &= -\frac{1}{2}\mathbf{H}\mathbf{c0}^T - \frac{1}{2}\mathbf{H0c}^T + \mathbf{HBH} = \mathbf{B}\end{aligned} \tag{3.25}$$

Algoritmo 3.14: Procedimiento del MDS Clásico

1. Calcular la matriz de disimilaridades $\Delta^{(2)}$
2. Aplicar la matriz doblemente centrada a esta matriz:

$$\mathbf{B}_\Delta = -\frac{1}{2}\mathbf{H}\Delta^{(2)}\mathbf{H}$$

3. Calcular la eigendescomposición $\mathbf{Q}\Lambda\mathbf{Q}^T$ de $\mathbf{B}_\Delta$
4. Sea *m* la dimensión de la solución. Denotar la matriz de los primeros *m* eigenvalores mayores que cero como Λ_+ y $\mathbf{Q}_+$ las primeras *m* columnas de $\mathbf{Q}$. Entonces, la matriz de coordenadas está dada por:

$$\mathbf{X} = \mathbf{Q}_+\Lambda_+^{1/2}$$

Si Δ es una matriz de distancias Euclidianas, entonces el MDS Clásico encuentra las coordenadas de una rotación. Nótese que la solución de $\mathbf{X} = \mathbf{Q}_+\Lambda_+^{1/2}$ es una solución sobre los ejes principales. En el paso 4, se pueden arrojar eigenvalores negativos pero no si Δ es una matriz de distancias Euclidianas.

Los primeros dos elementos son cero debido a que centrar un vector de unos produce un vector de ceros ($\mathbf{1}^T\mathbf{H}=\mathbf{0}$). Las matrices centradoras alrededor de $\mathbf{B}$ pueden descartarse debido a que se asume que las columnas de $\mathbf{X}$ tienen una media igual a cero. La operación en (3.25) se llama *doblemente centrada*. Para encontrar las coordenadas MDS de $\mathbf{B}$, es necesario factorizar $\mathbf{B}$ mediante la eigendescomposición. $\mathbf{Q}\Lambda\mathbf{Q}^T = (\mathbf{Q}\Lambda^{1/2})(\mathbf{Q}\Lambda^{1/2})^T = \mathbf{X}\mathbf{X}^T$.

El método del MDS clásico sólo difiere de este procedimiento en que la matriz de distancias al cuadrado $\mathbf{D}(\mathbf{X})^2$ es reemplazada por las disimilaridades al cuadrado $\Delta^{(2)}$. El procedimiento para el MDS clásico se muestra en el Algoritmo 3.14.

Una propiedad interesante del MDS clásico es que las dimensiones están anidadas. Esto significa que, por ejemplo, las primeras dos dimensiones de una solución MDS de tres dimensiones son exactamente las mismas que la de una solución MDS en dos dimensiones.

Una propiedad más del MDS es que en la matriz de coordenadas $\mathbf{X}$, la suma de los elementos de las columnas es cero. Esto significa que el origen de la configuración de $\mathbf{X}$ coincide con el centro de gravedad (centroide) de todos los puntos. Matemáticamente, se expresa de la siguiente forma

$$\sum_{r=1}^{n} x_{ri} = 0 \quad (i = 1, ..., m) \tag{3.26}$$

Un Ejemplo Numérico del MDS Clásico

A manera de ejemplo, emplearemos las distancias de las ciudades de la Tabla 3.1. Aquí, solo consideraremos las primeras cuatro. Así

$$\Delta = \begin{bmatrix} 0 & 569 & 667 & 530 \\ 569 & 0 & 1212 & 1043 \\ 667 & 1212 & 0 & 201 \\ 530 & 1043 & 201 & 0 \end{bmatrix}$$

entonces

$$\Delta^2 = \begin{bmatrix} 0 & 323761 & 444889 & 280900 \\ 323761 & 0 & 1468944 & 1087849 \\ 444889 & 1468944 & 0 & 40401 \\ 280900 & 1087849 & 40401 & 0 \end{bmatrix}$$

el segundo paso es calcular $\mathbf{B}_\Delta = -\frac{1}{2}\mathbf{H}\Delta^{(2)}\mathbf{H}$

$$\begin{aligned} \mathbf{B}_\Delta &= -\frac{1}{2}\begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{bmatrix}\begin{bmatrix} 0 & 323761 & 444889 & 280900 \\ 323761 & 0 & 1468944 & 1087849 \\ 444889 & 1468944 & 0 & 40401 \\ 280900 & 1087849 & 40401 & 0 \end{bmatrix} \\ &\quad \begin{bmatrix} \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{3}{4} \end{bmatrix} \\ &= \begin{bmatrix} 34466 & 101461 & -74893 & -61034 \\ 101461 & 492217 & -358045 & -235633 \\ -74893 & -358045 & 260637 & 172301 \\ -61034 & -235633 & 172301 & 124366 \end{bmatrix} \end{aligned}$$

El tercer paso es realizar la eigendescomposición de $\mathbf{B}_\Delta$

$$\mathbf{Q} = \begin{bmatrix} -0.500 & 0.439 & -0.728 & -0.161 \\ -0.500 & -0.313 & 0.318 & -0.741 \\ -0.500 & -0.654 & -0.171 & 0.540 \\ -0.500 & 0.529 & 0.581 & 0.362 \end{bmatrix}$$

y

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1.44e+02 & 0 & 0 \\ 0 & 0 & 2.13e+04 & 0 \\ 0 & 0 & 0 & 8.90e+05 \end{bmatrix}$$

Hay tres eigenvalores positivos. Esto implica que a lo más se puede obtener una representación en tres dimensiones. Ahora, se deben seleccionar los dos eigenvalores mas grandes de Λ (o tres, si se desea una representación en tres dimensiones). Estos dos eigenvalores formarán la diagonal

de la matriz Λ_+. Así, se observa que el eigenvalor mas grande se encuentra en la cuarta columna y el segundo eigenvalor en la tercer columna, por esto, de la matriz **Q** se debe seleccionar la cuarta y tercer columna de la matriz respectivamente para formar la matriz Q_+. Entonces, la matriz de coordenadas **X** se encuentra como sigue

$$\begin{aligned}\mathbf{X} &= \mathbf{Q}_+\Lambda_+^{1/2} \\ &= \begin{bmatrix} -0.161 & -0.728 \\ -0.741 & 0.318 \\ 0.540 & -0.171 \\ 0.362 & 0.581 \end{bmatrix} \begin{bmatrix} 9.43e+02 & 0 \\ 0 & 1.46e+02 \end{bmatrix} = \begin{bmatrix} -152.028 & -106.422 \\ -700.028 & 46.445 \\ 509.854 & -24.995 \\ 342.202 & 84.972 \end{bmatrix}\end{aligned}$$

Finalmente, nótese que las suma de los elementos de las columnas de la matriz de coordenadas **X** es igual a cero. Esto implica se satisface la ecuación (3.26) y por tanto, el origen del sistema **X** es el centroide de la configuración de todos los puntos.

3.3.7. Complejidad del MDS Clásico

En el Algoritmo 3.14, si la descomposición espectral que requiere el MDS Clásico se realiza con el método de las potencias, entonces es claro que la mayor carga computacional del algoritmo MDS Clásico corresponde al paso dos del algoritmo, que implica la multiplicación $\mathbf{B}_\Delta = -\frac{1}{2}\mathbf{H}\Delta^{(2)}\mathbf{H}$, la cual, corresponde a la multiplicación de tres matrices de $n \times n$ que toma $2n^3$ pasos. Sin embargo, esta multiplicación se realiza una sola vez al inicio del algoritmo. Esto implica que la carga computacional mayoritaria en el algoritmo recae en la descomposición espectral, por lo que se concluye que la complejidad del MDS Clásico es:

$$T_{MDSC} = O(i \times n^3) \tag{3.27}$$

Donde i es el número máximo de iteraciones necesarias para estimar los eigenvalores y eigenvectores empleando el método de las potencias (véase la sección 3.3.4).

3.3.8. Algoritmo SMACOF

SMACOF es un algoritmo iterativo alternativo para calcular el MDS sin necesidad de una descomposición espectral [8, 13]. Este método se basa en el algoritmo de *mayorización*. SMACOF es el acrónimo: Scaling by MAjorizing a COmplicated Function (Escalamiento mediante mayorización de una función complicada). Antes de presentar el algoritmo SMACOF, el algoritmo de mayorización se describe brevemente a continuación.

Algoritmo de Mayorización

El algoritmo de mayorización trata de minimizar una función complicada $f(x)$ empleando una función auxiliar $g(x, y)$. Esta función auxiliar debe seleccionarse de tal manera que para cada x en

el dominio de f se cumpla que $f(x) \leqslant g(x,y)$. Para una y particular en el dominio de g también se cumple que $f(y) = g(y,y)$.

Así, en las gráficas de f y g, la función g siempre está sobre la función f y g toca a f en el punto $x = y$. Así, la función g es una función de mayorización de f. La minimización de f puede realizarse mediante un esquema iterativo. En primer lugar, un valor inicial x_0 se usa para comenzar la minimización. Ahora, se define una función de mayorización adecuada $g(x, x_0)$. Esta función de minimización tiene su mínimo en, digamos, x_1. Ahora este valor de x define la función de mayorización $g(x, x_1)$ cuyo mínimo es x_2. Este proceso se repite hasta la convergencia.

A manera de ejemplo (Figura 3.5), supóngase que se desea minimizar la función f donde

$$\begin{aligned} f &: [-1.5, 2.0] \to \mathbb{R} \\ f(x) &= 6 + 3x + 10x^2 - 2x^4 \end{aligned}$$

Una función de mayorización g es seleccionada como

$$(x,y) \mapsto 6 + 3x + 10x^2 - 8xy^3 + 6y^4$$

el valor inicial para el algoritmo es x_0 =1.4, dando

$$g(x, 1.4) = 29.0496 - 18.952x + 10x^2$$

El mínimo de esta función es x =0.948. Entonces x_1 =0.948. La siguiente iteración da

$$g(x, 0.948) = 10.8460 - 3.7942x + 10x^2$$

El mínimo de esta función es x_2, y el proceso continúa hasta que el mínimo de $g(x,y)$ converge al mínimo de f.

Para obtener una solución MDS a partir del algoritmo de mayorización, se define la función de estrés como

$$S = \sum_{r<s} w_{rs}(\delta_{rs} - d_{rs})^2, \tag{3.28}$$

donde w_{rs} son pesos[4], δ_{rs} son disimilaridades y d_{rs} son las distancias Euclidianas calculadas a partir de las coordenadas de $\mathbf{X}$. Siguiendo a [13]

$$\begin{aligned} S &= \sum_{r<s} w_{rs}\delta_{rs}^2 + \sum_{r<s} w_{rs}d_{rs}^2(\mathbf{X}) - 2\sum_{r<s} w_{rs}\delta_{rs}d_{rs}(\mathbf{X}) \\ &= \eta_\delta^2 + \eta^2(\mathbf{X}) - 2\rho(\mathbf{X}) \end{aligned}$$

La función de estrés es reescrita en su forma matricial.

[4] w_{rs} es 1 si δ_{rs} es conocido y 0 si no.

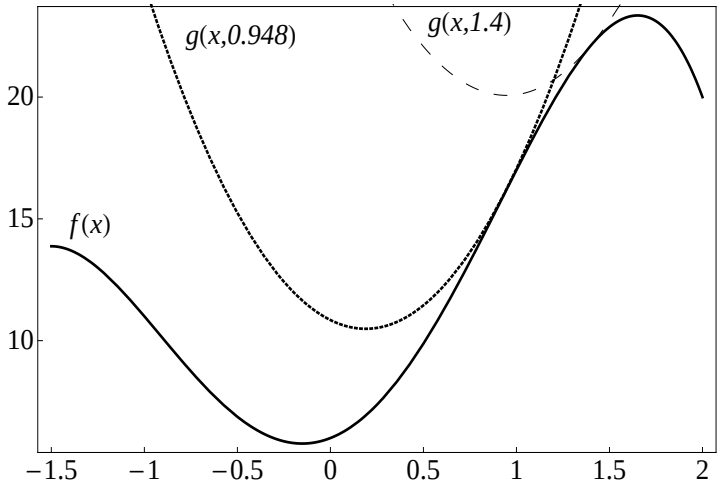


Figura 3.5: Minimización de la función $f(x) = 6 + 3x + 10x^2 - 2x^4$ empleando la función de mayorización $g(x, 1.4) = 29.0496 - 18.952x + 10x^2$ (primera iteración) y $g(x, 0.948) = 10.8460 - 3.7942x + 10x^2$ (segunda iteración).

$$\eta^2(\mathbf{X}) = \sum_{r<s} w_{rs}(x_r - x_s)^T(x_r - x_s) = \text{tr}(\mathbf{X}^T\mathbf{V}\mathbf{X}),$$

donde

$$v_{rr} = \sum_{r\neq s} w_{rs}; \qquad v_{rs} = -w_{rs} \quad (r \neq s).$$

Ahora

$$\begin{aligned}\rho(\mathbf{X}) &= \sum_{r<s} \frac{w_{rs}\delta_{rs}}{d_{rs}} d_{rs}^2 \\ &= \sum_{r<s} \frac{w_{rs}\delta_{rs}}{d_{rs}} (x_r - x_s)^T(x_r - x_s) \\ &= \text{tr}(\mathbf{X}^T\mathbf{B}(\mathbf{X})\mathbf{X}),\end{aligned}$$

Donde

$$b_{rs} = \begin{cases} w_{rs}\delta_{rs}/d_{rs}(\mathbf{X}) & \text{si} \quad d_{rs}(\mathbf{X}) \neq 0 \\ 0 & \text{si} \quad d_{rs}(\mathbf{X}) = 0 \end{cases}$$

Entonces la función de estrés se escribe como

$$S(\mathbf{X}) = \eta_\delta^2 + \text{tr}(\mathbf{X}^T\mathbf{V}\mathbf{X}) - 2\,\text{tr}(\mathbf{X}^T\mathbf{B}(\mathbf{X})\mathbf{X}) \tag{3.29}$$

Una función de mayorización T, para el estrés S está dada por

$$T(\mathbf{X}, \mathbf{Y}) = \eta_\delta^2 + \text{tr}(\mathbf{X}^T\mathbf{V}\mathbf{X}) - 2\,\text{tr}(\mathbf{X}^T\mathbf{B}(\mathbf{Y})\mathbf{Y}) \tag{3.30}$$

Algoritmo SMACOF para Mayorización del Estrés

Combinando las ecuaciones (3.29) y (3.30) se obtiene la desigualdad para la función de estrés.

$$\begin{aligned} S(\mathbf{X}) &= \eta_\delta^2 + \text{tr}(\mathbf{X}^T\mathbf{V}\mathbf{X}) - 2\,\text{tr}(\mathbf{X}^T\mathbf{B}(\mathbf{X})\mathbf{X}) \\ &\leqslant \eta_\delta^2 + \text{tr}(\mathbf{X}^T\mathbf{V}\mathbf{X}) - 2\,\text{tr}(\mathbf{X}^T\mathbf{B}(\mathbf{Y})\mathbf{Y}) = T(\mathbf{X},\ \mathbf{Y}) \end{aligned}$$

Así, $T(\mathbf{X}, \mathbf{Y})$ es una función de mayorización del estrés que es cuadrática en $\mathbf{X}$. El valor mínimo de esta función se obtiene analíticamente igualando su derivada a cero.

$$\frac{\partial T}{\partial \mathbf{Y}} = 2\mathbf{V}\mathbf{X} - 2\mathbf{B}(\mathbf{Y})\mathbf{Y} = 0$$

Así, $\mathbf{V}\mathbf{X} = \mathbf{B}(\mathbf{Y})\mathbf{Y}$. Para resolver este sistema de ecuaciones es necesario despejar $\mathbf{X}$, sin embargo, la inversa $\mathbf{V}^{-1}$ no existe debido a que $\mathbf{V}$ no es de rango completo. Empleando la inversa de Moore-Penrose para resolver esta ecuación se tiene

$$\mathbf{X} = \mathbf{V}^{+}\mathbf{B}(\mathbf{Y})\mathbf{Y} \tag{3.31}$$

donde $\mathbf{V}^{+} = (\mathbf{V} + \mathbf{1}\mathbf{1}^T)^{-1} - n^{-2}\mathbf{1}\mathbf{1}^T$. Sin embargo, el término $-n^{-2}\mathbf{1}\mathbf{1}^T$ es irrelevante para el algoritmo SMACOF. Ahora, si todos los pesos $w_{rs} = 1$, entonces la función de actualización es

$$\mathbf{X} = n^{-1}\mathbf{B}(\mathbf{Y})\mathbf{Y} \tag{3.32}$$

SMACOF se presenta en el Algoritmo 3.16.

3.3.9. Complejidad del Algoritmo SMACOF

Como se muestra en el Algoritmo 3.16, el algoritmo consiste, principalmente, en la multiplicación iterativa de las matrices $\mathbf{B}(\mathbf{Y})$ y $\mathbf{Y}$, donde $\mathbf{B}(\mathbf{Y})$ es una matriz de $n \times n$ y $\mathbf{Y}$ es una matriz de $n \times p$, donde p es la dimensión de los puntos. Es posible notar que la mayor carga computacional en este bucle corresponde a la multiplicación de dichas matrices. Esta multiplicación es de una matriz $n \times n$ por una de $n \times p$, la cual, requiere a lo más $n^2 \times p$ pasos. Por lo tanto, la complejidad del algoritmo SMACOF es $O(j \times (n^2 \times p))$, donde j es el número máximo de iteraciones y p la dimensionalidad de la solución. Sin embargo, debido a que la dimensión p es dos o tres y $p \ll n$ sin pérdida de generalidad, entonces la complejidad es

$$T_{SMACOF} = O(j \times n^2) \tag{3.33}$$

Algoritmo 3.16: Algoritmo SMACOF

1. Sea $\mathbf{Y} = \mathbf{X}^{[0]}$, donde $\mathbf{X}^{[0]}$ es la configuración de posiciones iniciales de los n puntos.
   `// Las posiciones iniciales pueden ser aleatorias o no`
   Establecer j=0.
2. Calcular el estrés $S(\mathbf{X}^{[0]})$, $S(\mathbf{X}^{[0]}) = \sum_{r<s} w_{rs}(\delta_{rs} - d_{rs}(\mathbf{X}^{[0]}))^2$
3. Incrementar j en uno.
4. Calcular la actualización $\mathbf{X}^{[j]} = n^{-1}\mathbf{B}(\mathbf{Y})\mathbf{Y}$
5. Calcular el estrés $S(\mathbf{X}^{[j]}) = \sum_{r<s} w_{rs}(\delta_{rs} - d_{rs}(\mathbf{X}^{[j]}))^2$
6. Si $S(\mathbf{X}^{[j-1]}) - S(\mathbf{X}^{[j]}) < \varepsilon$ o j =número máximo de iteraciones, Termina.
7. Sea $\mathbf{Y} = \mathbf{X}^{[j]}$ y regresar al paso 3.

3.3.10. Algoritmo MDS Clásico+SMACOF

El algoritmo SMACOF no sólo resuelve el MDS, sino que también puede emplearse como una herramienta de minimización del error del MDS Clásico. Esto es, una vez que se resuelve el MDS empleando el algoritmo MDS Clásico, el resultado —*la matriz de posiciones*— se emplea como entrada del algoritmo SMACOF para comenzar la minimización del error.

La combinación de los dos algoritmos es posible gracias a que el algoritmo SMACOF es capaz de resolver el MDS a partir de una matriz de coordenadas arbitraria, así, si dicha matriz contiene las coordenadas de los puntos, generadas por el MDS Clásico, entonces el SMACOF tratará resolver el MDS nuevamente, esto implica que la solución se ajustará una vez más hasta que las coordenadas de los puntos satisfagan lo mas posible a las disimilaridades. Este paso es equivalente a la minimización del error.

En el Algoritmo 3.17 se presenta la combinación de los algoritmos.

3.3.11. Complejidad del Algoritmo MDS Clásico+SMACOF

Es evidente que la complejidad del algoritmo está dada por la suma de los dos algoritmos ya que este nuevo algoritmo ejecuta, en primer lugar, el MDS Clásico y posteriormente el SMACOF. Por lo que la complejidad del algoritmo MDS Clásico+SMACOF está dada por:

$$T_{MDSC\_SMACOF} = O(i \times n^3) + O(j \times n^2) \tag{3.34}$$

donde i es el número máximo de iteraciones para realizar la eigendescomposición (Sec. 3.3.7) y j es el número máximo de iteraciones para completar el SMACOF (Sec. 3.3.9).

Es importante mencionar que el número de iteraciones j no será demasiado grande, ya que previamente el algoritmo MDS Clásico habrá encontrado una solución MDS aproximada. Debido a

Algoritmo 3.17: Procedimiento del MDS Clásico+SMACOF

1. Calcular la matriz de disimilaridades $\Delta^{(2)}$
2. Aplicar la matriz doblemente centrada a esta matriz:
$$\mathbf{B}_\Delta = -\frac{1}{2}\mathbf{H}\Delta^{(2)}\mathbf{H}$$
3. Calcular la eigendescomposición $\mathbf{Q}\Lambda\mathbf{Q}^T$ de $\mathbf{B}_\Delta$
4. Sea m la dimensión de la solución. Denotar la matriz de los primeros m eigenvalores mayores que cero como Λ_+ y $\mathbf{Q}_+$ las primeras m columnas de $\mathbf{Q}$. Entonces, la matriz de coordenadas está dada por:
$$\mathbf{X}^{[0]} = \mathbf{Q}_+\Lambda_+^{1/2}$$
5. Sea $\mathbf{Y} = \mathbf{X}^{[0]}$, donde $\mathbf{X}^{[0]}$ es la configuración de posiciones obtenidas mediante el MDS Clásico.
Establecer j=0.
6. Calcular el estrés $S(\mathbf{X}^{[0]})$, $S(\mathbf{X}^{[0]}) = \sum_{r<s} w_{rs}(\delta_{rs} - d_{rs}(\mathbf{X}^{[0]}))^2$
7. Incrementar j en uno.
8. Calcular la actualización $\mathbf{X}^{[j]} = n^{-1}\mathbf{B}(\mathbf{Y})\mathbf{Y}$
9. Calcular el estrés $S(\mathbf{X}^{[j]})$, $S(\mathbf{X}^{[j]}) = \sum_{r<s} w_{rs}(\delta_{rs} - d_{rs}(\mathbf{X}^{[j]}))^2$
10. Si $S(\mathbf{X}^{[j-1]}) - S(\mathbf{X}^{[j]}) < \varepsilon$ o j =número máximo de iteraciones, Termina.
11. Sea $\mathbf{Y} = \mathbf{X}^{[j]}$ y regresar al paso 7.

esto, el número de iteraciones requeridas para completar el SMACOF disminuye considerablemente. En contraste, cuando se ejecuta el SMACOF a partir de una matriz de coordenadas aleatoria, el algoritmo requiere de mas iteraciones para aproximar la solución MDS.

3.3.12. Isometrías

Como se ha mostrado en las secciones anteriores, las posiciones de todos los puntos del sistema obtenidos con el MDS, son las posiciones relativas entre cada par de puntos. Esto implica que, en primer lugar, la reconstrucción del sistema **X** mediante el MDS es una solución válida, ya que el sistema reconstruido está expresado en un sistema de coordenadas relativo al original. En otras palabras, puede que la reconstrucción de **X** resulte en un sistema rotado, reflejado o incluso, trasladado a un origen diferente con respecto a la solución real ya que, lo que importa es la configuración de todos los puntos. Así, las coordenadas relativas pudieran requerir (si así se desea) un arreglo final, es decir, una serie de transformaciones "cosméticas"para que la reconstrucción de la solución sea similar a la solución real. Este tipo de transformaciones pueden ser la *rotación*, *la traslación* y *la reflexión*. Debido a que este tipo de transformaciones dejan las distancias invariantes, se les suele decir *movimientos rígidos* o *isometrías*, su definición matemática [6] se enuncia a continuación

Definición 1 *Una transformación* ***T preserva la longitud*** *si para cada par de vectores* $\boldsymbol{X}_1$ *y* $\boldsymbol{X}_2$ *se cumple que*

$$||T(\boldsymbol{X}_2) - T(\boldsymbol{X}_1)|| = ||\boldsymbol{X}_2 - \boldsymbol{X}_1||$$

Una transformación que preserva la longitud es una ***isometría***

A continuación, se presentan de manera sintetizada las transformaciones lineales necesarias para ajustar la solución del MDS a la solución real. Para una descripción más amplia consultar [4, 6].

Proyección

Si **U** es un vector arbitrario, entonces, la proyección $\mathbf{P_U}$ de **X** sobre la línea del origen a través de **U** es[5]

$$\mathbf{P_U}(\mathbf{X}) = \frac{(\mathbf{X} \cdot \mathbf{U})\mathbf{U}}{(\mathbf{U} \cdot \mathbf{U})} \tag{3.35}$$

Ejemplo. Encontrar la proyección de $\mathbf{B} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ sobre la línea $\mathbf{A} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$ (Figura 3.6A)).

Así

$$\mathbf{P_A}(\mathbf{B}) = \frac{(\mathbf{A} \cdot \mathbf{B})\mathbf{A}}{(\mathbf{A} \cdot \mathbf{A})} = \frac{5}{10}\mathbf{A} = \frac{1}{2}\mathbf{A} = \begin{pmatrix} 3/2 \\ 1/2 \end{pmatrix}$$

[5]La proyección no es una isometría puesto que no preserva distancias, sin embargo este concepto es útil para explicar la reflexión.

Reflexión

Dado $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$, se desea encontrar las coordenadas del punto $S(\mathbf{X})$ tal que el punto medio del segmento de $\mathbf{X}$ a $S(\mathbf{X})$ es la proyección de $S(\mathbf{X})$ sobre la línea $\mathbf{U}$ que pasa por el origen, las coordenadas de $S(\mathbf{X})$ se denotan como x' y y'. Entonces, $\frac{1}{2}(\mathbf{X} + S(\mathbf{X})) = P(\mathbf{X})$, donde P es (3.35). Así

$$\mathbf{X} + S(\mathbf{X}) = 2P(\mathbf{X})$$

y

$$S(\mathbf{X}) = 2P(\mathbf{X}) - \mathbf{X} \tag{3.36}$$

Por ejemplo, cuando $\mathbf{U} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ de la ecuación (3.35), se obtiene

$$\begin{aligned} S(\mathbf{X}) &= \begin{pmatrix} x' \\ y' \end{pmatrix} = 2\begin{pmatrix} \frac{x+2y}{5} \\ \frac{2(x+2y)}{5} \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} \\ &= \begin{pmatrix} \frac{2(x+2y)}{5} - x \\ \frac{4(x+2y)}{5} - y \end{pmatrix} \end{aligned}$$

entonces

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -\frac{3}{5}x + \frac{4}{5}y \\ \frac{4}{5}x + \frac{3}{5}y \end{pmatrix}$$

Figura 3.6B)

Por ejemplo, si $\mathbf{X} = \begin{pmatrix} 0 \\ 10 \end{pmatrix}$, entonces, $S(\mathbf{X}) = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 8 \\ 6 \end{pmatrix}$ cuando $\mathbf{U} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

Traslación

La *suma* de dos vectores es la suma de sus componentes, así, para $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$ y $\mathbf{U} = \begin{pmatrix} u \\ v \end{pmatrix}$, tenemos

$$\mathbf{X} + \mathbf{U} = \begin{pmatrix} x + u \\ y + v \end{pmatrix}$$

De manera general, se puede obtener una interpretación geométrica de la suma de vectores. Sea un triángulo con vértices $\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} x \\ 0 \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix}$ y lo movemos de tal manera que su primer vértice se

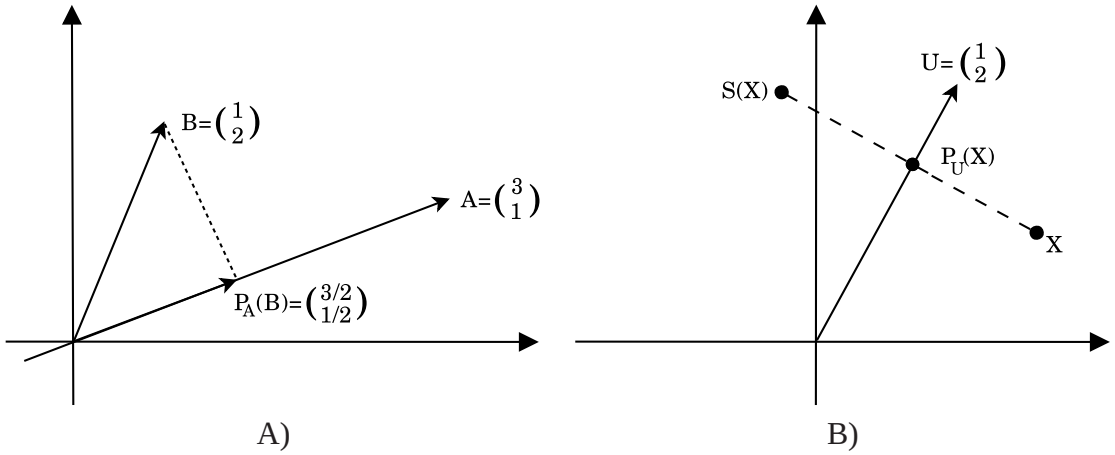


Figura 3.6: A) Proyección de B sobre A. B) Reflexión de X sobre la línea U.

encuentre en $\begin{pmatrix} u \\ v \end{pmatrix}$, entonces los otros dos vértices están en $\begin{pmatrix} x+u \\ v \end{pmatrix}$ y $\begin{pmatrix} x+u \\ u+y \end{pmatrix}$, respectivamente. Véase la Figura 3.7 A).

Entonces, la suma de los vectores $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$ y $\mathbf{U} = \begin{pmatrix} u \\ v \end{pmatrix}$ se obtiene trasladando el segmento dirigido de $\mathbf{0}$ a $\mathbf{X}$ de manera paralela a él mismo hasta que el origen se encuentre en $\mathbf{U}$.

Rotación

Se define la transformación R_θ como la rotación de θ radianes como sigue. Sea $\mathbf{X}$ un vector. Rotar el segmento de $\mathbf{0}$ a $\mathbf{X}$ alrededor de $\mathbf{0}$ en sentido de las manecillas del reloj un ángulo de θ radianes.

Sea $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$ y $R_\theta(\mathbf{X}) = \begin{pmatrix} x' \\ y' \end{pmatrix}$. Se reescribe $\mathbf{X}$ y $R_\theta(\mathbf{X})$ en la forma

$$\mathbf{X} = |X| \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix}, \; R_\theta(\mathbf{X}) = |R_\theta(\mathbf{X})| \begin{pmatrix} \cos\phi' \\ \sin\phi' \end{pmatrix}$$

donde ϕ es el ángulo polar de $\mathbf{X}$ y ϕ' es el ángulo polar de $R_\theta(\mathbf{X})$. Entonces $\phi' = \phi + \theta$ y $|R_\theta(\mathbf{X})| = |\mathbf{X}|$

$$R_\theta(\mathbf{X}) = |\mathbf{X}| \begin{pmatrix} \cos(\theta+\phi) \\ \sin(\theta+\phi) \end{pmatrix} = \begin{pmatrix} |\mathbf{X}|\cos\phi\cos\theta - |\mathbf{X}|\sin\phi\sin\theta \\ |\mathbf{X}|\cos\phi\cos\theta + |\mathbf{X}|\sin\phi\sin\theta \end{pmatrix}$$

Ahora,

$$|\mathbf{X}|\cos\phi = x, \; |\mathbf{X}|\sin\phi = y$$

Entonces

$$R_\theta(\mathbf{X}) = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ y\cos\theta + x\sin\theta \end{pmatrix} \tag{3.37}$$

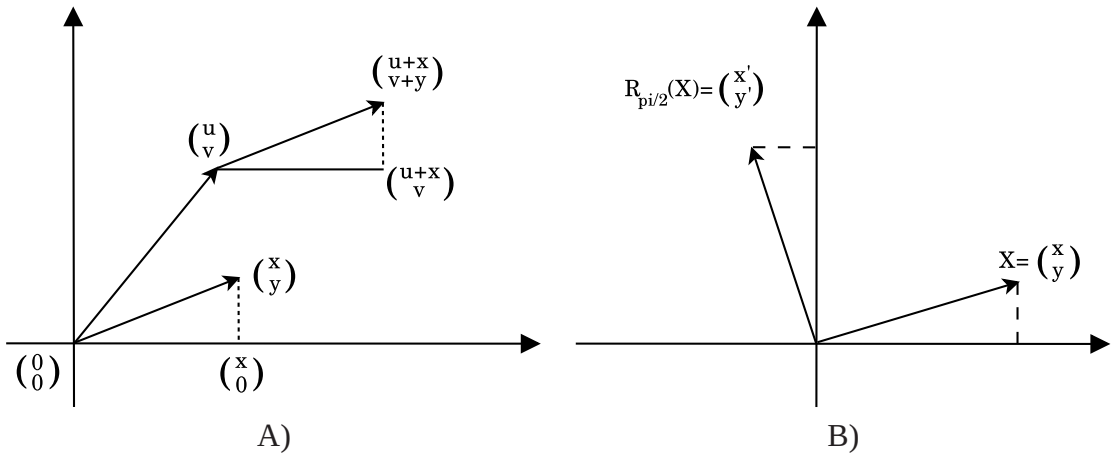


Figura 3.7: A) Traslación del triángulo con vértices en $(0, 0)$, $(x, 0)$ y (x, y) al punto (u, v). B) Rotación de X $\pi/2$ radianes.

Ejemplo (Figura 3.7B)). Sea $\mathbf{X} = \begin{pmatrix} x \\ y \end{pmatrix}$, calcular $R_{\pi/2}(X)$. Entonces, de (3.37) tenemos que

$$R_{\pi/2}(\mathbf{X}) = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x\cos(\pi/2) - y\sin(\pi/2) \\ y\cos(\pi/2) + x\sin(\pi/2) \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}$$

Ejemplo

Volviendo al ejemplo de las diez ciudades europeas. Puede observarse que la configuración de las ciudades no tiene la posición convencional que se encuentra en un mapa geográfico del continente Europeo, es claro que, Roma no es la ciudad que se encuentra más al norte. Sin embargo, es posible emplear una transformación isométrica, o incluso, una combinación de ellas para ajustar la configuración arrojada por el MDS. En este caso, únicamente se requiere rotar la configuración resultante de tal manera que Estocolmo sea la ciudad que se encuentra más al norte y más al este con respecto al resto de las ciudades. En la Figura 3.8 se muestran dos configuraciones, la reconstrucción original obtenida con el MDS al igual que la configuración de las ciudades después de la rotación.

Figura 3.8: A) Reconstrucción de las 10 ciudades europeas con el MDS. B)Rotación de la configuración de las diez ciudades Europeas para ajustarse a una posición convencional.

Capítulo 4
Algoritmo de Localización UAM-MDS

En este capítulo se presenta el algoritmo de localización UAM Multidimensional Scaling (UAM-MDS), el cual emplea tres herramientas fundamentales: el algoritmo de partición de Awerbuch; el algoritmo Bellman-Ford y finalmente, el MDS con las isometrías necesarias para ajustar los mapas. El algoritmo propuesto requiere únicamente la información de conectividad de cada uno de los nodos, es decir, quién está en el radio de comunicaciones de quién. Comenzando con una red dada, se ejecuta, en primer lugar, el algoritmo de partición de Awerbuch para crear varias colonias con una densidad de integrantes uniforme para cada uno de ellas. Enseguida, empleando el algoritmo Bellman-Ford, cada líder de colonia calcula una matriz de distancias en saltos, entre cada par de sensores de su colonia. La matriz de distancias es la matriz de disimilaridades requerida por el MDS para generar mapas con posiciones relativas de cada una de las colonias que componen la red (Figura 4.1).

Cuando cada colonia cuenta con su mapa relativo, entonces el algoritmo mezcla los mapas, a manera de un *rompecabezas*, suponiendo que cada colonia cuenta con tres faros. Entonces se crea un mapa único con coordenadas globales y el algoritmo termina.

4.1. Algoritmo General

Algoritmo 4.18: UAM-MDS. Algoritmo General

Entrada: Una WSN modelada con una gráfica de disco unitario $G = (V, E)$ y hay una arista $\{u, v\} \in E$ si y solo si la distancia Euclidiana entre u y v menor o igual que 1.

Resultado: Los nodos que conforman la red conocen su posición aproximada en un sistema global de coordenadas.

Datos: Cada nodo conoce quiénes son sus vecinos inmediatos (los que están a los más a distancia 1).

1. Dada una red $G = (V, E)$. Particionar la red empleando el Algoritmo de partición de Awerbuch mediante el sincronizador γ. Cada colonia creada cuenta con un coordinador de operaciones llamado *líder de colonia*.

2. En cada colonia, cada integrante transmite su lista de vecinos sobre el árbol que conforma la colonia. Cuando la información converge en el líder de colonia, se crea la matriz de adyacencias local **A** y ejecuta el Algoritmo Bellman-Ford sobre esta matriz para crear la matriz de distancias **D**, la cual, contiene las mejores trayectorias, en saltos, para cada par de sensores pertenecientes a la colonia.

3. Cada líder de colonia aplica el MDS sobre su matriz de distancias (disimilaridades) **D** para generar un mapa local de cada una de las colonias. La longitud que se le asigna al salto se selecciona según el número de habitantes y la densidad de la colonia. Las longitudes adecuadas del salto se obtuvieron experimentalmente y pueden encontrarse en la Sección 5.3 y en el Apéndice A.

4. Cuando cada colonia termina de calcular el MDS y, suponiendo que cuenta con al menos tres nodos faro, entonces usa la mejor isometría (rotación, traslación y reflexión) para transformar sus coordenadas relativas en un sistema de coordenadas globales.

5. Finalmente, una vez que cada colonia se ajusta a su posición absoluta, toda la red se encuentra en un sistema global absoluto y cada nodo conoce su posición, así, el algoritmo termina.

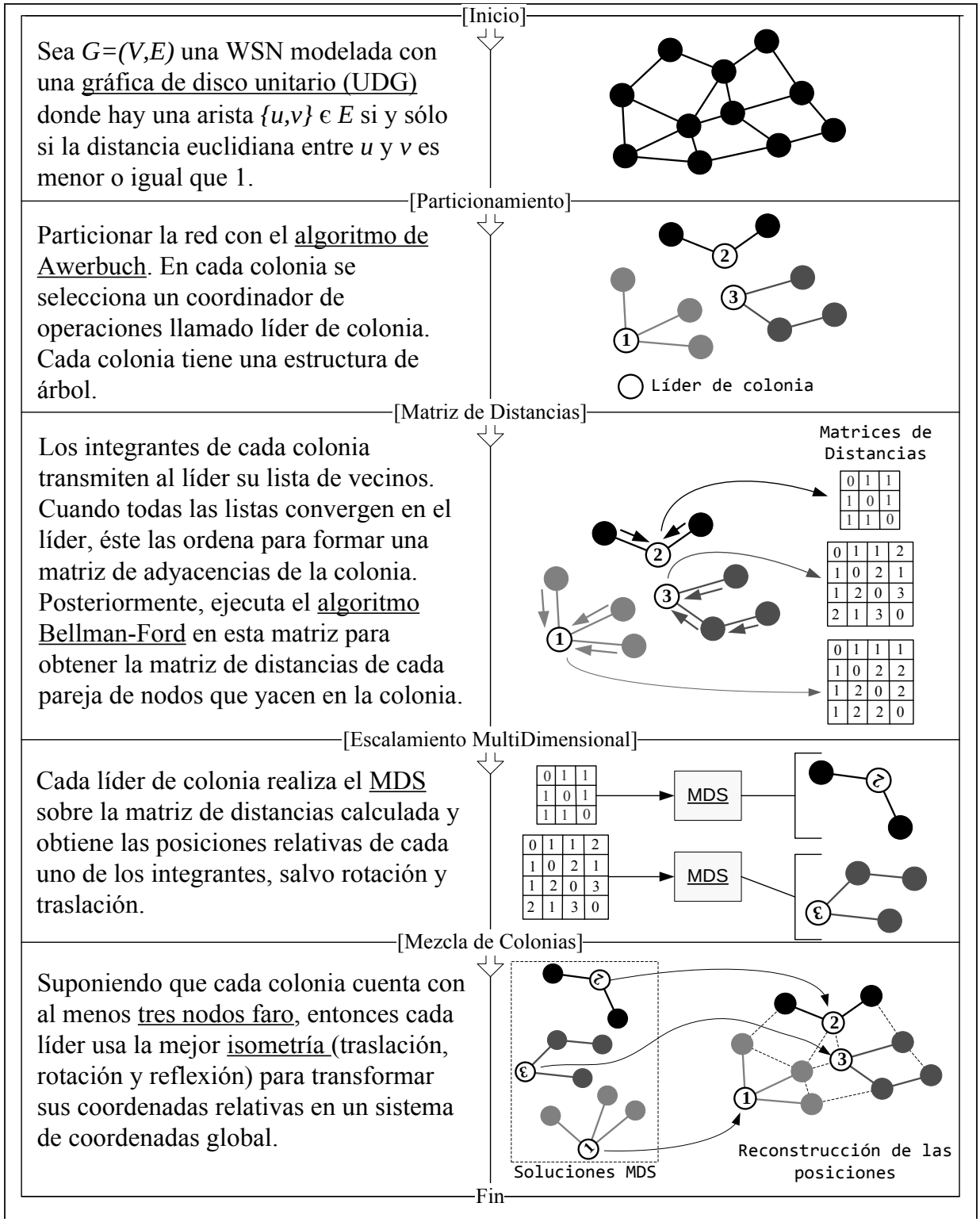


Figura 4.1: Representación gráfica del algoritmo de localización.

4.1.1. Particionamiento

El algoritmo de partición de Awerbuch es eficiente en tiempo y en comunicaciones. Además, el algoritmo construye las colonias como un máximo subconjunto de nodos cuyo diámetro no excede el logaritmo de su cardinalidad. Esto garantiza que el número total de colonias vecinas es lineal y el diámetro máximo de la colonia es el logaritmo del número de nodos de la red. Esto implica que cada colonia se crea de tal manera que crece rápidamente. Siempre y cuando cada capa nueva que se agregará a la colonia contenga al menos k veces el número de nodos que fueran agregados anteriormente.

En el algoritmo MDS-MAP(P) [27], se emplea un enfoque de partición, en el cual, cada nodo es un líder de colonia. La colonia se forma con los vecinos del líder que se encuentran a no más de dos saltos de distancia. Los mapas locales se mezclan basándose en los nodos que tienen en común, de acuerdo con la mejor transformación lineal que mezcla las coordenadas de los nodos de un mapa a otro. Por esto, si la red se compone de n nodos, el MDS se ejecuta n veces sobre cada colonia. Esto implica que el MDS se ejecuta demasiadas veces de manera innecesaria. En cambio, el algoritmo de Awerbuch crea las colonias con un diámetro que puede parametrizarse para acotar los costos del MDS. Por otro lado, sólo el líder calcula el MDS en su colonia, esto permite reducir costos de ejecución del algoritmo.

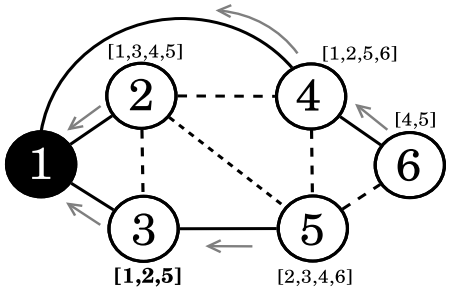
MDS-MAP(P) realiza n cálculos de MDS con un solo fin: lograr que cada mapa local creado cuente con varios nodos comunes con algún otro mapa para mezclarlos correctamente, sin embargo, durante la construcción de las particiones con el algoritmo de Awerbuch, se crea una infraestructura capaz de proporcionar nodos comunes entre colonias vecinas donde el número de nodos comunes es el mínimo posible y el necesario para mezclar los mapas.

4.1.2. Mejores Rutas Intra-Colonias

Para la construcción de las posiciones relativas, MDS requiere trabajar sobre una matriz de distancias que relaciona a cada par de nodos sensores. Inicialmente, cada nodo dentro de la red conoce únicamente la información de conectividad disponible, esto es, cada nodo cuenta con una lista de vecinos a un salto de distancia. Al finalizar el algoritmo de partición, cada colonia forma un árbol de búsqueda en amplitud BFS (Sección 3.1.2). Entonces, cada nodo tiene conocimiento de la infraestructura de la colonia y es capaz de distinguir que nodos contiguos pertenecen a la colonia así como también quién es su nodo padre y sus nodos hijos. Con esta información, cada nodo crea una lista de vecinos que pertenecen a la misma colonia. Cuando el nodo completa su lista de vecinos, la información se transmite hacia el líder de colonia ascendiendo por el árbol hasta llegar a la raíz. Cuando la información converge en el líder, se ordena y crea la matriz de adyacencias. Figura 4.2 A) y B).

Empleando el algoritmo Bellman-Ford (Sección 3.2), el nodo líder genera la matriz con la mejor trayectoria (en saltos) para cada par de nodos (Figura 4.2 C)). Esta matriz resultante será la matriz de disimilaridades que se empleará en el MDS.

En esta etapa del algoritmo, cada nodo líder se encarga de ejecutar el algoritmo Bellman-Ford, cuya complejidad en operaciones es $O(|V|^3)$. Sin embargo, la propagación de la información de las listas crea un gasto en comunicaciones.



A)

	1	2	3	4	5	6
1	0	1	1	1	∞	∞
2	1	0	1	1	1	∞
3	1	1	0	∞	1	∞
4	1	1	∞	0	1	1
5	∞	1	1	1	0	1
6	∞	∞	∞	1	1	0

B)

	1	2	3	4	5	6
1	0	1	1	1	2	2
2	1	0	1	1	1	2
3	1	1	0	2	1	2
4	1	1	2	0	1	1
5	2	1	1	1	0	1
6	2	2	2	1	1	0

C)

Figura 4.2: A) Colonia de seis nodos. La línea continua es la estructura de la colonia (árbol BFS). Cada nodo envía su lista de vecinos al líder de colonia (nodo 1). B) A partir de la información recibida, el líder crea la matriz de adyacencias. C) Matriz de distancias con las mejores rutas, en saltos, creada con el algoritmo Bellman-Ford.

Cada nodo que compone la colonia transmite su lista de vecinos hacia el líder de colonia sobre la estructura de árbol creada por el algoritmo de partición, este proceso se repite hasta que la información converge en el líder. Así, durante la propagación se envían $O(|V|^2)$ mensajes.

Por tanto, se concluye que esta etapa del algoritmo tiene la complejidad

$$\begin{aligned} C_{BF} &= O(|V|^2) \\ T_{BF} &= O(|V|^3). \end{aligned} \tag{4.1}$$

en comunicaciones y en tiempo, respectivamente.

4.1.3. Posiciones Relativas de la Colonia

En el momento que el nodo líder completa la matriz de disimilaridades comienza el tercer paso del algoritmo de localización: construir los mapas locales de cada colonia. Para esta tarea, cada líder estima las posiciones de los integrantes de la colonia mediante el algoritmo del MDS Clásico (Sección 3.3.6) o mediante el algoritmo SMACOF (Sección 3.3.8) para crear sus propios mapas locales, la complejidad de sendos algoritmos es $O(i \times n^3)$ y $O(j \times n^2)$, respectivamente.

El resultado del MDS es un mapa relativo que proporciona la posición de los nodos dentro de la colonia. Estas posiciones son relativas de un nodo con respecto a los demás, y el origen de la

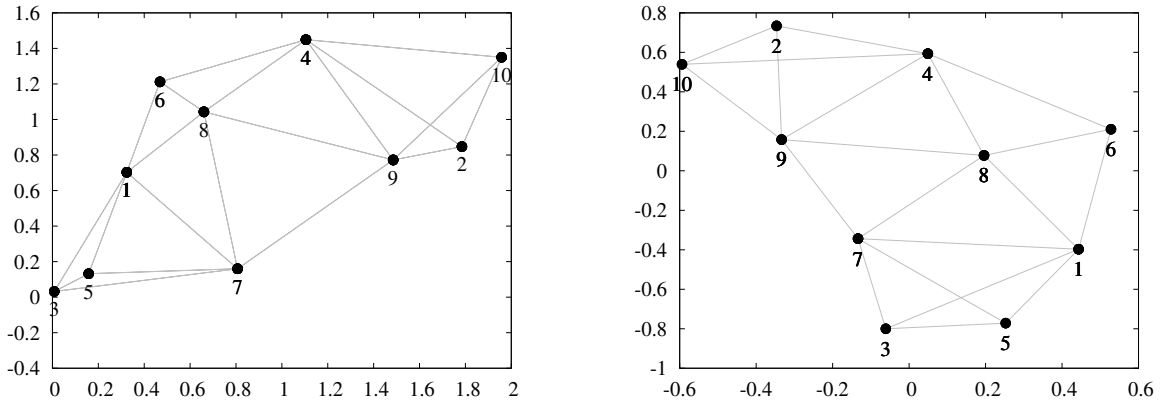


Figura 4.3: A) Red de sensores con posiciones reales. B) Red de sensores reconstruida con el MDS con información de conectividad (posiciones relativas).

configuración del MDS es el centro de gravedad de los puntos. En la Figura 4.3 A) se observa una red con sus posiciones originales. Cuando se reconstruyen las posiciones originales con el MDS, tal como se observa en la Figura 4.3 B), puede notarse, por un lado, que el origen de la configuración es el centro de gravedad y, por otro lado, es necesario ajustar la configuración resultante para que se parezca el original, con alguna isometría por ejemplo.

4.1.4. Uso de Faros para la Mezcla de Colonias y Obtención de las Coordenadas Absolutas

Como se presentó en la sección 2.2, algunas contribuciones emplean los nodos faro como puntos de referencia que son indispensables para completar la localización, es decir, el algoritmo de localización no podría finalizar correctamente sin la asistencia de los faros. En contraste, un algoritmo de localización que emplea como motor de funcionamiento el MDS, no requiere de los faros sino hasta la última etapa del algoritmo, esto es, cuando el MDS reconstruye las posiciones de cada nodo, entonces los faros se emplean para crear un mapa con posiciones absolutas.

Una solución MDS requiere únicamente de tres o cuatro faros, tres para una reconstrucción 2D y cuatro para 3D. Por ejemplo, supóngase que una reconstrucción MDS puede verse como un *mapa* que presenta las posiciones de los nodos. Este mapa, tiene su origen en el centro de gravedad del conjunto de los puntos y, además, las posiciones individuales de cada nodo, no necesariamente se asemejan a las posiciones reales ya que el MDS produce una configuración de puntos arbitraria, sin embargo, las distancias entre cada par de nodos corresponden lo mejor posible a las distancias reales. Así, los faros ajustan la solución MDS de tal manera que la orientación de la solución sea lo más parecida a las posiciones reales. Esta tarea se realiza empleando de manera ordenada las siguientes isometrías: traslación, rotación y reflexión. En la Figura 4.4A), se presenta una reconstrucción MDS de cinco nodos en un plano Euclidiano de dos dimensiones, nótese que el origen de la configuración es el centro de gravedad de los puntos. Ahora bien, supóngase que los faros,

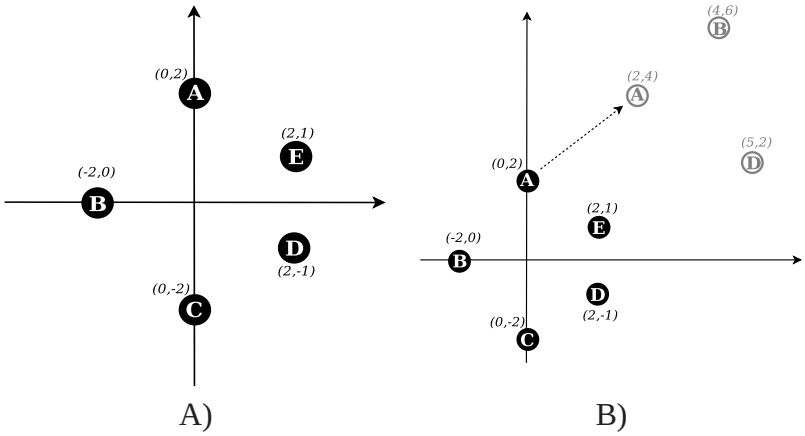


Figura 4.4: A) Reconstrucción MDS, B) Traslación de toda la solución MDS al punto (2, 4).

que conocen su propia posición, son los nodos A, B y D como se muestra en la Figura 4.4B). El primer paso para el ajuste de la reconstrucción consiste en seleccionar un faro como un punto de referencia, por ejemplo, el faro A. Después, toda la solución MDS se *traslada* de tal manera que la posición del nodo A en (0,2) de la reconstrucción MDS corresponda a la posición real del faro, es decir, en la posición (2,4).

Una vez que la solución se traslada, como se muestra en la Figura 4.5A), el nodo *A* se encuentra en la posición real: (2,4), esto implica que por ninguna razón este nodo podrá moverse de su posición, esto es equivalente a colocar un *clavo* en la posición (2,4) del mapa MDS. Ahora, sólo resta mover el segundo nodo a su posición real para continuar ajustando la solución MDS, para esto, se selecciona el nodo *B*. Debido a que el nodo *A* ahora está fijo, el mapa MDS únicamente se puede rotar o reflejar pero no trasladar, así, por conveniencia se selecciona la rotación.

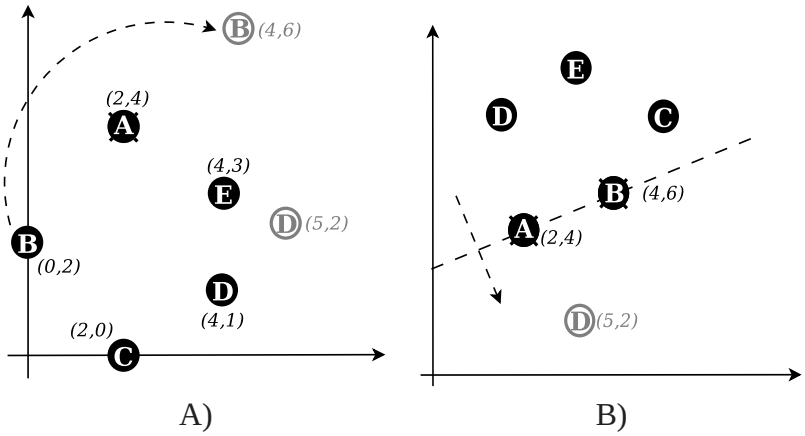


Figura 4.5: A) Rotación del sistema MDS al punto B(4, 6), con el faro *A* como *pivote*, B)reflexión de la solución MDS para ajustar al sistema al punto D(5, 2), sobre la línea que forma A y B.

La rotación se realiza sobre todo el mapa reconstruido. Para esto, se emplea el nodo *A* como *pivote* ya que este nodo se encuentra fijo. De esta manera, se debe rotar toda la solución MDS hasta

que el nodo *B* se ajuste a su posición real (4,6), tal y como se muestra en la Figura 4.5B).

Finalmente, ahora que los nodos *A* y *B* ya se encuentran en su posición real, la única isometría disponible es la reflexión de la reconstrucción MDS sobre la línea que forman los nodos *A* y *B*. Por esto, se requiere un tercer faro (nodo *D*) para ajustar la reconstrucción del mapa con la mejor reflexión sobre la línea que forman los dos nodos. La Figura 4.6 muestra la reconstrucción MDS orientada en su posición convencional.

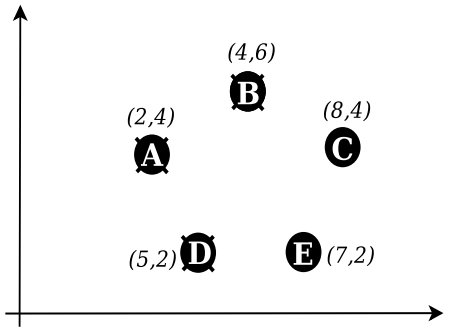


Figura 4.6: Solución MDS ajustada a la posición real.

Si una solución MDS se construye en tres dimensiones es necesario emplear cuatro faros. Esto es, al emplear alguna isometría es necesario considerar las tres coordenadas de las posiciones de los nodos. Así, el cuarto faro permite ajustar la solución MDS, por ejemplo, moviendo el mapa a través del eje *Z* en el plano.

Cuando se trabaja bajo un enfoque de particionamiento, cada colonia resuelve su propio MDS y el resultado es el conjunto de posiciones relativas de cada nodo. Ahora bien, si se considera que cada colonia cuenta con al menos tres faros, entonces la colonia puede ajustarse a su posición absoluta final. Si se realiza esta acción para cada colonia que conforma la red, entonces esto es equivalente a reconstruir un *rompecabezas*, donde cada colonia es una *pieza* y sus respectivas posiciones y orientación, obtenidas mediante los faros, son equivalentes a las *hendiduras* de la pieza. Esto hace a cada pieza única y por tanto, implica que tenga una sola posición en el plano, al igual que sus vecinas únicas conectadas a ella. Una vez que cada colonia se ajusta a su posición final, cada uno de los nodos conoce su posición absoluta y el algoritmo de localización termina. Este paso se realiza una vez que se resuelve el MDS en cada colonia, para lo cual se considera que existen tres faros en cada una.

Capítulo 5
Resultados

En esta sección se presenta el análisis de los resultados obtenidos mediante simulación. Esta sección está estructurada como sigue: en primer lugar, se presenta una una discusión acerca de las ventajas del particionamiento. Después, la comparación del tiempo y la cantidad de mensajes transmitidos entre los dos algoritmos de particionamiento, esto es, la versión original de Awerbuch y el algoritmo de partición modificado. Asimismo, se presenta el comportamiento del algoritmo Bellman-Ford para distintos tamaños de colonias y para distintas densidades de red. Con esto se pretende obtener la longitud del salto que minimice el error entre las distancias inter-nodo estimadas y las distancias reales de la red. Finalmente, se analizan y comparan los dos algoritmos que resuelven el MDS, estos son, MDS Clásico, SMACOF y la combinación de estos.

Todos los algoritmos fueron programados en un simulador de eventos discretos (DES) programado en C++ [20]. Este simulador permite el diseño y análisis de algoritmos distribuidos. La recolección y análisis de las estadísticas pudo realizarse gracias al empleo de las librerías de otro DES llamado SSS [24].

5.1. Particionamiento

El algoritmo de partición es la columna vertebral del algoritmo de localización, ya que el proceso de particionamiento de una red permite reducir la complejidad de los cálculos matemáticos que un sólo nodo debe realizar, puesto que se trabaja solo con una pequeña porción de nodos que componen la red. El algoritmo de partición de Awerbuch, al igual que el algoritmo de partición modificado, requieren de un análisis más exhaustivo para observar el comportamiento individual de ciertas variables, por ejemplo, la cantidad de mensajes individuales transmitidos por nodo o el factor de crecimiento de las colonias.

La idea principal de particionar la red es dividirla en distintas colonias para reducir la carga computacional de los nodos para realizar la localización. Esto es deseable debido a que la complejidad de instrucciones del MDS puede requerir hasta $O(i \times n^3)$ si se emplea MDS Clásico o $O(i \times n^2)$ si se emplea SMACOF. Así, al observar el orden de estas complejidades, es evidente que si se construyen varias colonias en una red, tanto la cantidad de instrucciones como los mensajes se reducen considerablemente debido al orden del crecimiento. La ganancia de emplear particionamiento se presenta en la Figura 5.1, donde se graficaron las complejidades de instrucciones y mensajes requeridas por el MDS Clásico para distintas redes particionadas y no particionadas. En

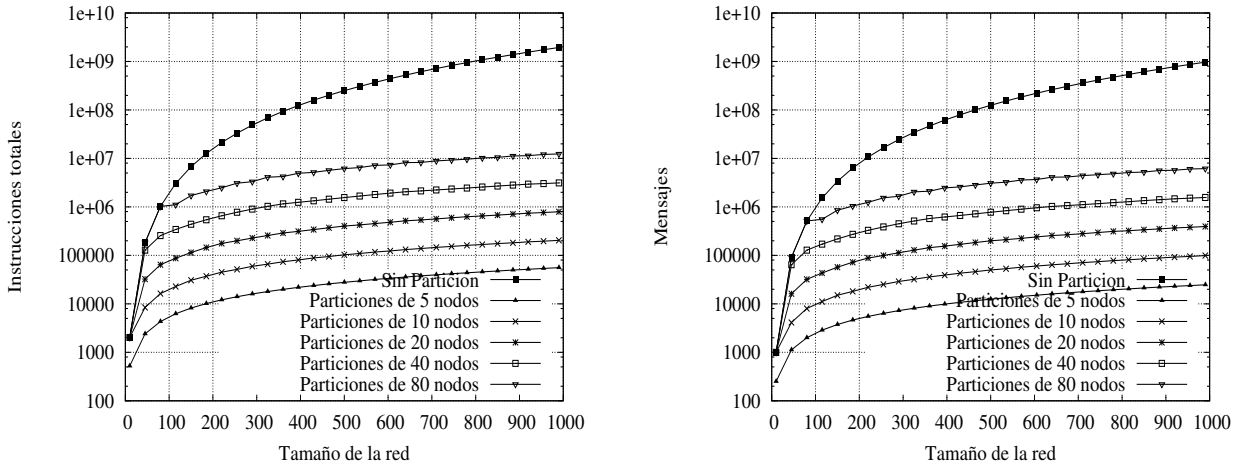


Figura 5.1: A) Comparación en la cantidad de instrucciones cuando se emplea el particionamiento y una red no particionada, B) comparación en la cantidad de mensajes transmitidos.

primera instancia, se nota que siempre que cualquier red pueda particionarse, las complejidades se reducen, comparados con el total de mensajes e instrucciones requeridas por una red no particionada. Sin embargo, supóngase que en una red de 80 nodos se ejecuta el algoritmo de partición y, por alguna razón, toda la red conforma una sola colonia, entonces aquí el particionamiento es un proceso que no genera ninguna ventaja ya que la cantidad de operaciones se incrementa por el sobreprecio del particionamiento sin producir el efecto deseado.

El número de elementos que conforman cada colonia es un factor determinante en la cantidad de instrucciones y mensajes requeridos por el MDS, esto es, cuando el tamaño de las colonias aumenta, las complejidades también aumentan. En la Figura 5.1 se observa claramente que las colonias compuestas por cinco nodos disminuyen los recursos computacionales y de comunicaciones, comparados otras colonias de mayor tamaño.

Cabe mencionar que ninguno de los algoritmos de particionamiento construye colonias de tamaño fijo, esta característica depende en gran parte del tamaño, la topología y la densidad de red. Así, independientemente del tamaño de las colonias, las operaciones y mensajes se verán considerablemente reducidos siempre que la red se particione.

5.1.1. Algoritmo de Partición de Awerbuch Modificado

En la Sección 3.1.4, se presenta una alternativa del algoritmo de partición de Awerbuch en la cual se detalla que la construcción de cada colonia no necesariamente debe ser secuencial. Con una modificación al algoritmo se puede conseguir que la partición de la red se realice de forma paralela, es decir, en lugar de seleccionar únicamente un líder para crear una nueva colonia, se seleccionan varios líderes que comienzan su construcción de manera simultánea.

Mediante simulación, es posible comparar estos criterios con más detalle para observar el comportamiento de ciertas variables, las cuales, pueden ser factor determinante en el desempeño del

Tabla 5.1: Comparación entre los criterios Secuencial y Paralelo

95 % Conf.	**Secuencial**			**Paralelo**		
Variables	$\overline{x}$	σ	e	$\overline{x}$	σ	e
Tiempo de Finalización	809.257	81.671	0.716	213.012	32.064	0.281
Total de Mensajes	28146.730	4622.685	40.520	25445.318	4858.834	42.59
Mensajes por nodo	49.772	8.462	0.074	45.813	6.77	0.059
Total de Mensajes por líder	1920.816	242.143	2.122	2091.704	267.214	2.342
Mensajes promedio por líder	111.794	11.238	0.099	83.073	6.771	0.059
Número de Colonias	4.370	2.090	0.018	11.980	3.461	0.030
Tamaño prom. colonia	19.336	4.397	0.039	11.113	3.334	0.029

algoritmo en una red inalámbrica de sensores. Esta evaluación se necesita debido a que el análisis de las complejidades únicamente proporciona, en notación asintótica, los límites superiores del consumo de recursos de tiempo y comunicaciones. Así, una simulación puede proporcionar tanto el comportamiento de los sensores en general, como el comportamiento de los nodos que fueron seleccionados como líderes de colonia.

La Tabla 5.1 muestra una comparación cuantitativa entre el criterio secuencial y el paralelo. Se realizaron 50,000 iteraciones para cada criterio. Durante cada simulación, se creó aleatoriamente una red de 600 nodos y se obtuvieron estadísticas de las siguientes variables: el tiempo en que finaliza el algoritmo, es decir, desde que inicia el algoritmo en $t = 0$ hasta que el último mensaje del algoritmo se recibe. Se asume que cada mensaje transmitido por el algoritmo tiene un retardo de una unidad de tiempo. También se presenta la cantidad de mensajes totales que se enviaron desde el inicio hasta la terminación del algoritmo, al igual que el promedio de mensajes transmitidos por nodo. Para comparar el trabajo ejercido por los líderes de colonia, la tabla muestra el total de mensajes que transmiten únicamente los líderes al igual que el promedio de mensajes transmitidos durante la ejecución del algoritmo. Finalmente, se presenta el número de colonias y su tamaño promedio. Los resultados se presentan con una confianza del 95 %.

Los resultados obtenidos muestran, en primera instancia, que el criterio paralelo termina, en promedio, en un tiempo menor que su similar secuencial. Por otra parte, también se observa que el algoritmo paralelo transmite menos mensajes que el algoritmo original de Awerbuch y además, también reduce de manera considerable la complejidad en tiempo y de comunicaciones, esto se concluye observando únicamente el total de mensajes transmitidos y el tiempo de terminación del algoritmo.

La disminución de los recursos de tiempo y de comunicaciones se debe a que, en el criterio paralelo, cuando inicia el algoritmo se crea la primer colonia, el líder transmite una solicitud a sus descendientes *hoja* para que envíen un *candidato,* el cual será el líder de una nueva colonia. Cada nodo responde al líder con un candidato. Por su parte, el líder los recibe y transmite la orden de generar las particiones con todos los candidatos. En el criterio secuencial sólo se selecciona un líder de todos los posibles, en cambio, en el algoritmo modificado se seleccionan varios a la vez y crea una lista que se transmite en un solo mensaje hacia los nodos hoja, lo cual provoca que varias colonias sean creadas casi de manera simultánea y así, el tiempo de terminación del algoritmo

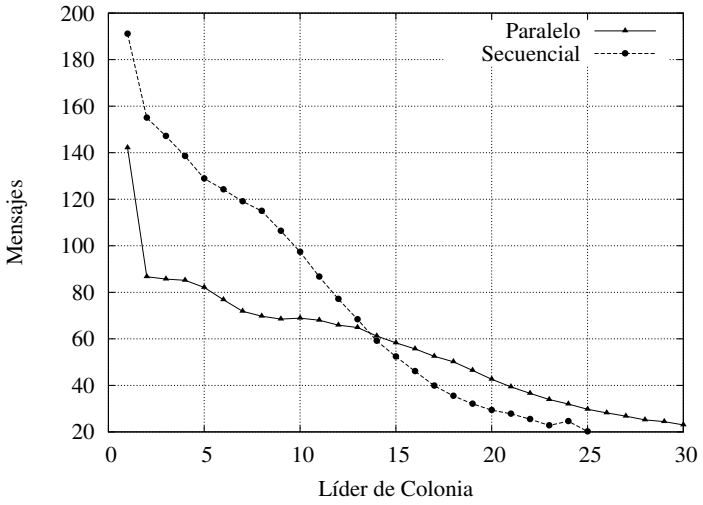


Figura 5.2: Mensajes transmitidos por los líderes en el orden en que cada uno de ellos fue creado.

disminuye. Por esto, la modificación del algoritmo implica reducir la cantidad de mensajes ya que, por ejemplo, la creación secuencial requiere mensajes extra para realizar los *rastreos en retrocesos* (backtracking) requeridos para la búsqueda del líder. Es claro que paralelizando la creación de las colonias este tipo de mensajes no son necesarios.

En la Figura 5.2 se muestran los mensajes promedio transmitidos por cada uno de los líderes, para ambos algoritmos. Se presentan los mensajes que transmite cada líder en el estricto orden en que cada uno de ellos fue elegido. Se observa que la cantidad de mensajes decrece en ambos ya que la gráfica residual se está reduciendo. En el criterio secuencial el promedio de los mensajes transmitidos por los líderes es mayor debido a los mensajes extra requeridos para el *backtracking*. En ambos criterios, inevitablemente los líderes que fueron creados al inicio del algoritmo transmiten mucho más mensajes que el resto. Esto de debe a que, cuando la primer colonia se crea, tiene la posibilidad de elegir uno o varios líderes de los nodos que se encuentran en la periferia de la colonia ya que ésta es la única creada y la gráfica residual aún no ha sido particionada. Conforme se crean nuevas colonias, los líderes tienen menos candidatos a seleccionar debido a que la gráfica residual se está reduciendo. Esto implica que el número de mensajes transmitidos por cada líder disminuye conforme avanza el algoritmo y por tanto, los líderes, cuyas colonias fueron creadas en las primeras etapas del algoritmo, transmiten mucho más mensajes que los elegidos en las etapas subsiguientes.

Ahora bien, de la Tabla 5.1 nótese que el número de colonias promedio creadas en ambos criterios difiere. Esta diferencia se debe exclusivamente al criterio empleado, es decir, cuando se emplea la paralelización los líderes *contienden* por los nodos que aún no han sido añadidos a alguna colonia. Debido a esto, las colonias no crecen a una razón uniforme, provocando la creación de más colonias, pero con un número menor de habitantes. En contraste, en el algoritmo secuencial los líderes no contienden por los nodos permitiendo que cada colonia *crezca* de manera uniforme y, crear menos colonias con un mayor número de habitantes. Así, la red particionada en forma secuencial tiene la característica principal de que las colonias son uniformes en cuanto a su número

de habitantes.

Finalmente, se pueden diferenciar las características esenciales de ambos criterios. El criterio secuencial mantiene un compromiso entre tiempo y comunicaciones y trata de distribuir la carga en comunicaciones tanto de los líderes como de los nodos *comunes*, por esto, el algoritmo permite crear menos colonias con un número uniforme de habitantes. En contraste, el criterio paralelo reduce considerablemente el tiempo de ejecución del algoritmo y además, mejora la distribución del trabajo realizado por los líderes. Por esto, seleccionar el criterio implica terminar el algoritmo de partición en un tiempo mucho menor, a costa de la distribución del trabajo y el número de elementos de cada colonia.

5.1.2. Criterio de Crecimiento de las Colonias

Independientemente del criterio empleado, el algoritmo de partición construye cada colonia como un subconjunto máximo de nodos cuyo diámetro no excede el logaritmo de su cardinalidad. Así, cada colonia se crea tal que la altura máxima del árbol BFS que conforma la colonia $H_P < \log_2 V / \log_2 k$, para un parámetro k, $2 \leq k \leq |V|$. Donde la elección de k significa un compromiso entre el tiempo y el número de mensajes. Entonces, k es el factor de crecimiento de las colonias, es decir, cada capa que se agrega a cierta colonia debe tener al menos k veces el número de elementos que los ya agregados anteriormente. Por esto, durante la creación, si cada capa que se agregará no satisface este criterio, entonces, se detiene el crecimiento de la colonia y continúa con la construcción de la siguiente.

La elección del valor de k depende, en gran parte, de la aplicación o del usuario. Sin embargo, el algoritmo de partición toma la decisión de continuar creando una colonia para cierto valor de k, siempre y cuando la densidad de la red sea adecuada. De esta manera, el algoritmo de partición Awerbuch sugiere que cada colonia se construya con un factor de crecimiento cuyos valores son $2 \leq k \leq |V|$. Los valores que se asignan a k son para que cada colonia crezca, si es posible, *exponencialmente*. Esto implica también que el algoritmo finalice en un tiempo menor. Sin embargo, emplear un valor de $0 < k \leq |V|$, implica que para los valores de $0 < k \leq 2$ las colonias no crecen exponencialmente sino que, por el contrario, la restricción de crecimiento se *relaja*, permitiendo que las colonias crezcan libremente sin muchas restricciones, pero el tiempo de finalización del algoritmo aumenta.

Para observar este efecto, obsérvese la red de sensores que se muestra en la Figura 5.3 A), donde B), C) y D) son tres resultados distintos para $k = 0.7$, $k = 1$ y $k = 2$, respectivamente. Por simplicidad, considérese únicamente la colonia identificada con el marcador "■".

Para $k = 1$, la colonia creada detiene su crecimiento cuando la última capa que pudiera agregarse no contiene el mismo número de elementos que el total de elementos agregados a la colonia previamente. Esto significa que la colonia ya es suficientemente grande y que la adición de más nodos no implica que continuará creciendo exponencialmente. Ahora, la elección de $k = 2$ implica que cada capa de la colonia debe contener dos veces el total de elementos ya agregados previamente. Por ejemplo, en la Figura 5.3 D), la colonia cuenta con 45 elementos y su altura máxima es de dos capas. Entonces, para agregar una tercera capa, ésta debe contener $2 \times 45 = 90$ elementos. Esta condición no se satisface debido a que la red no es tan densa para para permitir un crecimiento que

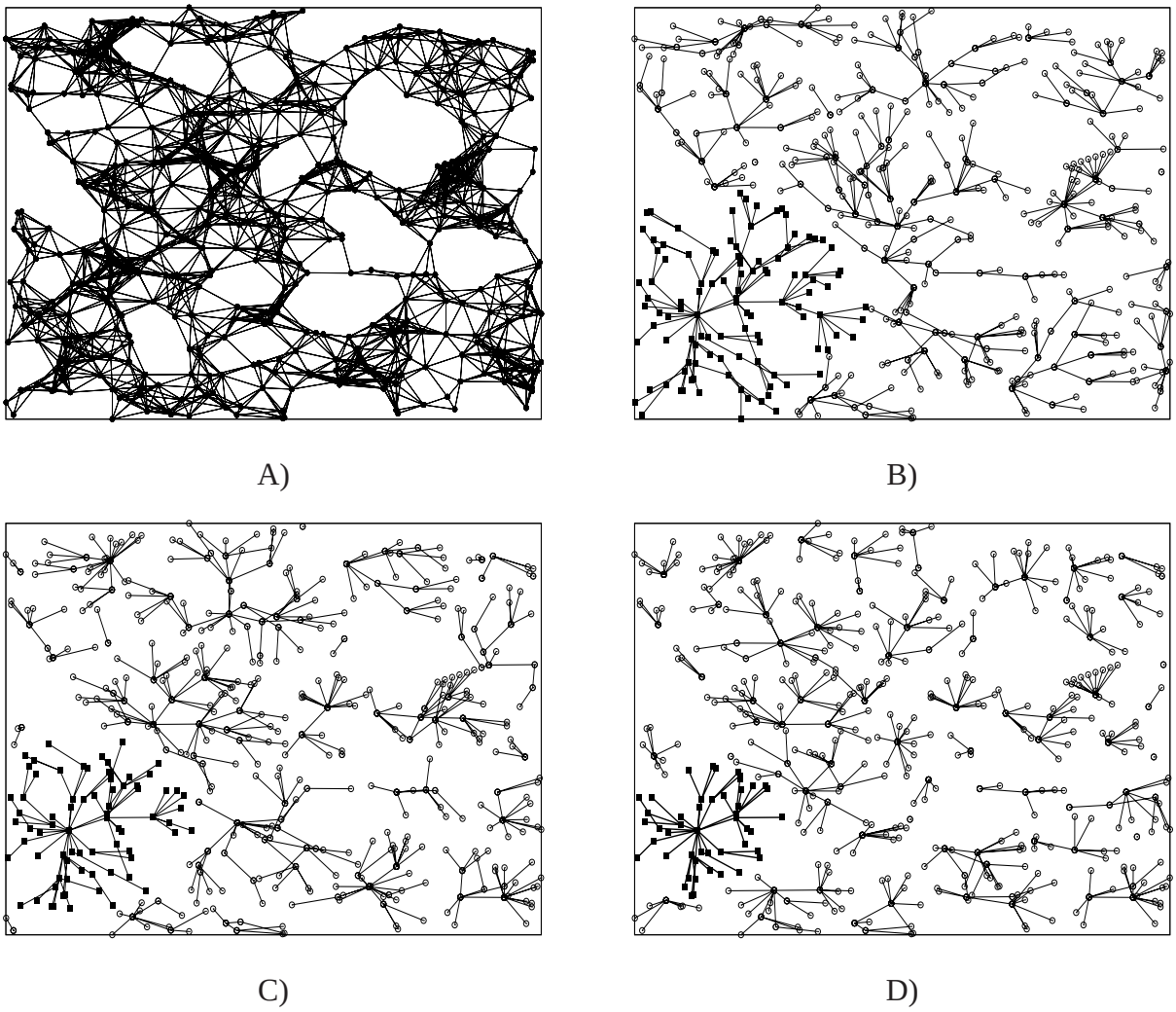


Figura 5.3: A) Red inalámbrica de 500 sensores. B) Construcción de particiones empleando una $k = 0.7$, C) $k = 1$ y D) $k = 2$.

involucre una $k = 2$. Por esta razón, el número de colonias creadas es mayor que al emplear una $k = 1$.

Finalmente, k puede tomar tomar cualquier otro valor. Si se emplea, por ejemplo k =0.7, implicaría que la condición de crecimiento es más *flexible* de tal manera que las colonias tienen más libertad de crecimiento, aunque la red sea rala. En la Figura 5.3 A) se muestra el resultado de emplear una k =0.7. Nótese que las colonias son muy grandes y, por lo tanto, el número de colonias creadas es menor, comparadas con $k = 1$ y $k = 2$.

El valor de k apropiado permite construir una colonia con el tamaño que se requiera, independiente de la densidad de la red.

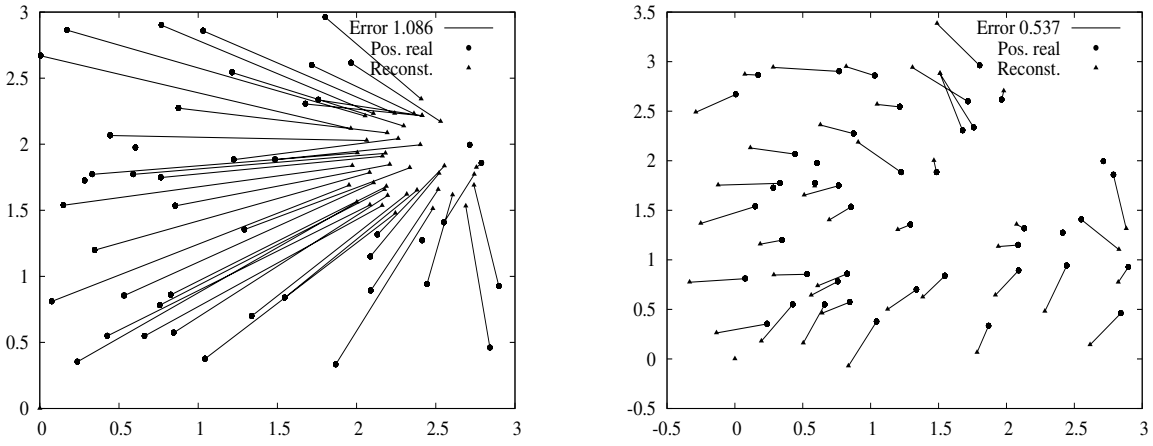
La complejidad del algoritmo de partición, tanto en la versión secuencial como en la versión paralela, es también una función del valor que se le asigna a k (Véase la sección 3.1.3).

5.2. Algoritmo Bellman-Ford

El algoritmo Bellman-Ford se emplea para crear la matriz de distancias que relaciona cada par de nodos dentro de la colonia. Sin embargo, es preciso definir el término *distancia* desde el punto de vista de un algoritmo libre de rango. En un algoritmo de este tipo la única información que dispone cada nodo es la identidad de sus vecinos inmediatos y no cuenta con ningún tipo de información acerca de la distancia Euclidiana entre ellos. Para estos esquemas, la distancia no es más que el número de *saltos* (hops) que es necesario dar para llegar de un nodo a otro.

Para un algoritmo de localización no es suficiente conocer el número de saltos porque la localización está basada en las distancias reales o aproximadas entre nodos. Por esto, es necesario cuantificar la longitud del salto. Por ejemplo, para un esquema basado en rango, si se conoce la distancia entre dos nodos, la longitud del salto será esta misma distancia. Por el contrario, para un esquema libre de rango, si dos nodos pueden comunicarse entre si, entonces la distancia entre ellos no es mayor que el radio de comunicaciones R del sensor, esto implica que a los más el valor del salto será R. Estas consideraciones dan libertad de seleccionar empíricamente el valor adecuado del salto, cuyo valor está entre 0 y R. El diseño del algoritmo UAM-MDS, implica utilizar un valor único para la red entera, a diferencia de [22, 31], que mediante el uso de suficientes faros, el valor del salto de cada enlace se estima mediante la obtención de promedios del vecindario. UAM-MDS no considera un enfoque similar, en cambio, mediante el análisis de las distancias reales y las distancias para diferentes valores del salto, propone un valor único que minimiza el error. En cada red donde se ejecuta el Bellman-Ford, variando las longitudes del salto, se encuentra una longitud del salto que sustituye todas las distancias reales entre cada pareja de nodos con cierto error. Por ejemplo, si el error promedio es 0.31 entonces la longitud del salto seleccionada experimentalmente aleja o acerca a todos los nodos un 31 % de la distancia real.

El principio de este análisis es el siguiente. Supóngase que la longitud de un salto es R. Esta condición es estricta en el siguiente sentido: para una red de sensores desplegada aleatoriamente, difícilmente todos las distancias entre un nodo y sus vecinos son R o están muy cerca de R. Entonces, por ejemplo, si la distancia real entre dos nodos es $0.3R$ y la longitud del salto considerada para construir la matriz de distancias es R, implica que los nodos se alejan 3.3 veces más de la distancia real. Este comportamiento es la motivación del análisis ya que la minimización del error se



A) B)

Figura 5.4: A) Error en la reconstrucción MDS de la una red empleando un salto con longitud 0.2*R*, B) reconstrucción con 0.8*R*.

verá reflejada en la precisión de la localización. La Figura 5.4 muestra la influencia de la longitud del salto en la reconstrucción de las posiciones cuando se emplea el MDS. Nótese que la elección adecuada de la longitud del salto minimiza, en promedio, el error en las distancias calculadas.

Una definición intuitiva de la *densidad* de una red de sensores es la cantidad de nodos sensores posicionados por unidad de área en un plano de dos dimensiones. Aquí, el término *área* permanece un tanto ambiguo debido a que el despliegue de la red de sensores puede ser aleatorio y la región de monitoreo puede ser desconocida. Sin esta información, la manera de estimar la densidad de la red es mediante la conectividad de los nodos sensores, es decir, la cantidad de vecinos promedio con los que cuentan los nodos pueden proporcionar la densidad para un área igual a la región en la cual un nodo sensor es capaz de comunicarse con sus vecinos. Así, la definición conveniente de densidad para este trabajo es: cantidad de nodos promedio posicionados por un área de conectividad con radio *R* (Figura 5.5).

La forma de seleccionar la longitud del salto es mediante experimentación. La idea general del experimento es la siguiente: generar aleatoriamente una red de sensores y estimar la distancia real entre cada par de nodos. Después, se encuentran las distancias entre cada par de nodos al ejecutar el algoritmo Bellman-Ford sobre esa misma red, considerando que la longitud del salto es 0.1*R*. Posteriormente, se estima el error cuadrático entre las distancias reales y las distancias obtenidas con el algoritmo. Este proceso se repite para los distintas longitudes del salto, desde 0.1*R* hasta 1.0*R* y siempre sobre la misma red. Cuando se han realizado los experimentos para todos los valores posibles del salto, se selecciona el valor cuyo error es el mínimo global de todos los experimentos. De esta manera, se selecciona la longitud conveniente para que las distancias estimadas con el Bellman-Ford se asemejen lo más posible a las distancias reales. Este procedimiento se repite para distintas redes generadas aleatoriamente compuestas por 10, 20, 30,...,130 nodos. Debido a que el algoritmo Bellman-Ford se ejecuta sobre cada colonia de manera independiente, cada red creada representa una colonia de alguna red particionada previamente, por ejemplo, una red de 20 nodos

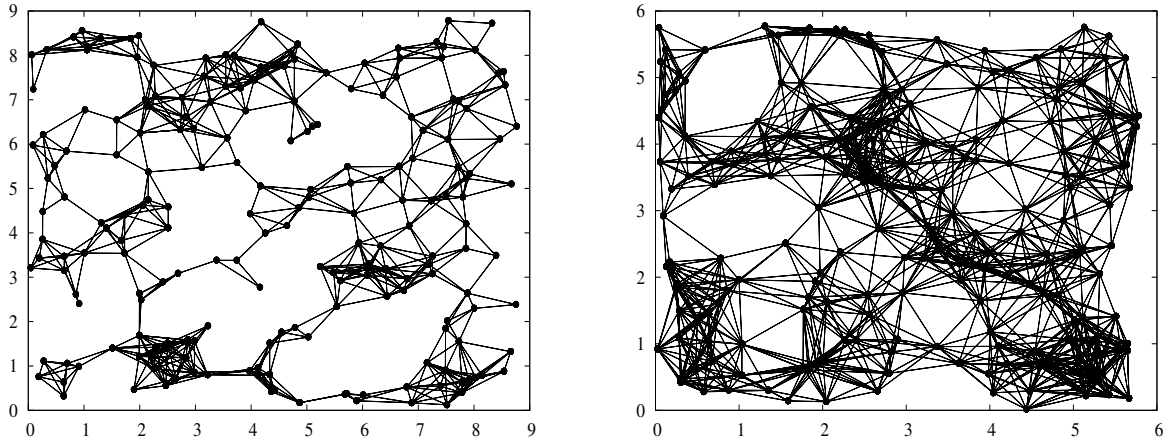


Figura 5.5: A) Red con una densidad promedio de $7.19\frac{nodos}{R}$. B)Red con una densidad promedio de $15.2\frac{nodos}{R}$.

será considerada una colonia de 20 nodos. Para evaluar el comportamiento de la longitud del salto cuando la densidad de la red cambia, cada red se construye con cuatro densidades distintas: 7.64, 10.78, 13.690 y $15.71\frac{nodos}{R}$.

La Tabla A.1 (Apéndice A) muestra los resultados obtenidos del experimento. Para cada tamaño de red se generaron 8,000 redes aleatorias distribuidas en las cuatro densidades dando como resultado 96,000 simulaciones en total.

Empíricamente se encontró un valor del salto que minimiza el error en las distancias intra-nodos para redes de tamaño y densidad diferente. No obstante, esto no significa que el error sea pequeño, pero tampoco que el error es el mismo para las densidades consideradas. Por ejemplo, en la Figura 5.6 se graficó el error promedio en las distancias estimadas para redes de diferente tamaño (de 10 a 130 nodos) para cada una de las cuatro densidades (7.64, 10.78, 13.690 y $15.71\frac{nodos}{R}$). Claramente se nota que, independientemente del tamaño de la red, el error promedio en las distancias se incrementa cuando la densidad de la red es baja y que disminuye cuando la densidad aumenta. Este comportamiento sugiere que el error es inversamente proporcional a la densidad de la red.

Este resultado no puede pasarse por alto debido a que el conocimiento de los factores que influyen en la minimización de las distancias, se reflejará en el resultado final de la localización ya que el algoritmo Bellman-Ford proporciona la entrada principal al sistema MDS: la matriz de distancias.

5.3. Escalamiento Multi-Dimensional

El análisis requerido para evaluar el MDS consiste, básicamente, en utilizar los resultados de las secciones anteriores. Esto es, en primer lugar, cada colonia se considera una red independiente compuesta de *n* nodos. Posteriormente, ejecutando el algoritmo Bellman-Ford se construye la ma-

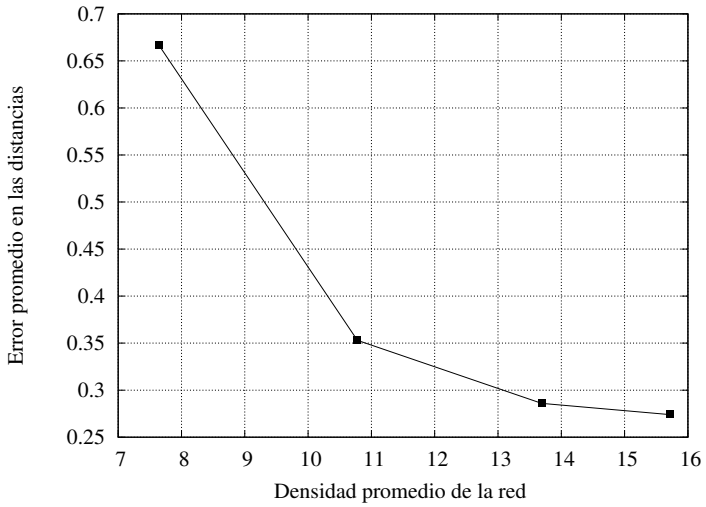


Figura 5.6: Influencia de la densidad de la red en el error de la estimación de las distancias internodo

triz de distancias que relaciona la cantidad de saltos que son necesarios realizar para llegar de un nodo a cualquier otro. La matriz de distancias se introduce al sistema MDS para reconstruir las posiciones de los nodos que componen la colonia. Sin embargo, recordando que el MDS reconstruye las posiciones relativas de los nodos en un plano Euclidiano, entonces, es necesario realizar una serie de arreglos *cosméticos* para que la reconstrucción se asemeje a la solución real. Esto es, se deben emplear un conjunto de isometrías (traslación, reflexión y rotación) para ajustar la solución MDS y de esta manera, poder compararla con las posiciones reales. El ajuste se realiza suponiendo que la colonia contiene tres faros, así, ajustando los tres faros a su posición real, toda la solución MDS se ajusta, hasta donde es posible, a las posiciones reales. Este procedimiento puede observarse en la Figura 5.7. Donde A) es una colonia compuesta de 70 sensores. Entonces, cuando se crea la matriz de distancias mediante Bellman-Ford se realiza el MDS. En B) se presenta el error en la posición entre las posiciones reales y las posiciones de la reconstrucción MDS, donde las posiciones MDS son las posiciones relativas de cada nodo. Nótese que aquí el error es muy grande ya que aún no se emplean los faros. Finalmente, con la ayuda de tres faros, la solución MDS se ajusta lo mejor posible a las posiciones reales, lo que provoca que el error disminuya considerablemente, tal y como se muestra en C).

El error en la reconstrucción MDS se debe a distintos factores. Por ejemplo, como se mencionó en la sección 2.2, la calidad de la reconstrucción depende tanto de la rigidez de la gráfica como de la hipótesis de que la trayectoria más corta entre dos nodos debe ser aproximadamente igual a la distancia Euclidiana entre ellos, tal hipótesis no necesariamente es cierta. También, tanto la elección de la longitud del salto durante el algoritmo Bellman-Ford como la densidad de la red produce un error considerable en la localización, ya que el valor elegido produce la matriz de distancias que posteriormente, se introduce al MDS. Este efecto se muestra en las Figuras 5.7 C) y D), donde una vez que los faros se emplean para ajustarse a las posiciones reales, se debe seleccionar la longitud adecuada del salto para que minimice el error en la reconstrucción. Así, C) es la reconstrucción

MDS empleando un salto de $0.8R$ y D) es la reconstrucción para un salto de $0.2R$. Nótese que la diferencia de error entre ambos es grande y depende, en gran parte, de la elección correcta del valor del salto durante el algoritmo Bellman-Ford.

Los resultados obtenidos en el experimento de la sección anterior (Tabla A.1) son esenciales para identificar que existe un valor del salto que minimiza el error en las distancias intra-nodos así como la influencia de las densidades en este resultado. La reconstrucción MDS está estrechamente ligada a estos resultados, sin embargo, para el MDS el error no es la diferencia entre las distancias reales y las distancias estimadas por el algoritmo Bellman-Ford, sino que desde ahora, el error medido será el de las posiciones estimadas contra las reales. No obstante, se espera que que el MDS tenga un comportamiento similar al algoritmo Bellman-Ford, en el cual el error en las posiciones disminuya conforme crece la densidad y viceversa.

Para medir el error en las posiciones, se realizó un experimento similar al de la sección anterior, el cual consiste en crear redes aleatoriamente de 10, 20,...,130 nodos con 4 distintas densidades: 7.64, 10.78, 13.690 y $15.71\frac{nodos}{R}$. En cada red se ejecuta el algoritmo Bellman-Ford, variando la longitud del salto de $0.1R$, $0.2R$... hasta R para construir la matriz de distancias. Para cada salto, se ejecuta el MDS con la asistencia de tres faros para ajustar la reconstrucción los más posible a la solución real. Una vez que se tiene este ajuste, se mide el error entre la reconstrucción MDS y las posiciones reales. Cuando se ejecuta el MDS para cada longitud del salto, y para cada densidad, se selecciona el salto que minimice el error promedio en la localización.

Tanto el MDS como el SMACOF requieren de una serie de operaciones matemáticas sobre matrices grandes; por ejemplo, se requieren multiplicaciones entre matrices, sumas, o incluso, la descomposición espectral de ellas. Sin embargo, para la implementación de las herramientas matemáticas que conforman sendos algoritmos es posible emplear un conjunto de librerías de alto nivel llamadas GNU Octave [12]. La ventaja de usar este software es que se dispone de todas las librerías necesarias que pueden ligarse en cualquier programa de C++, esto permite usar cualquier instrucción de Octave sobre el simulador de eventos discretos seleccionado.

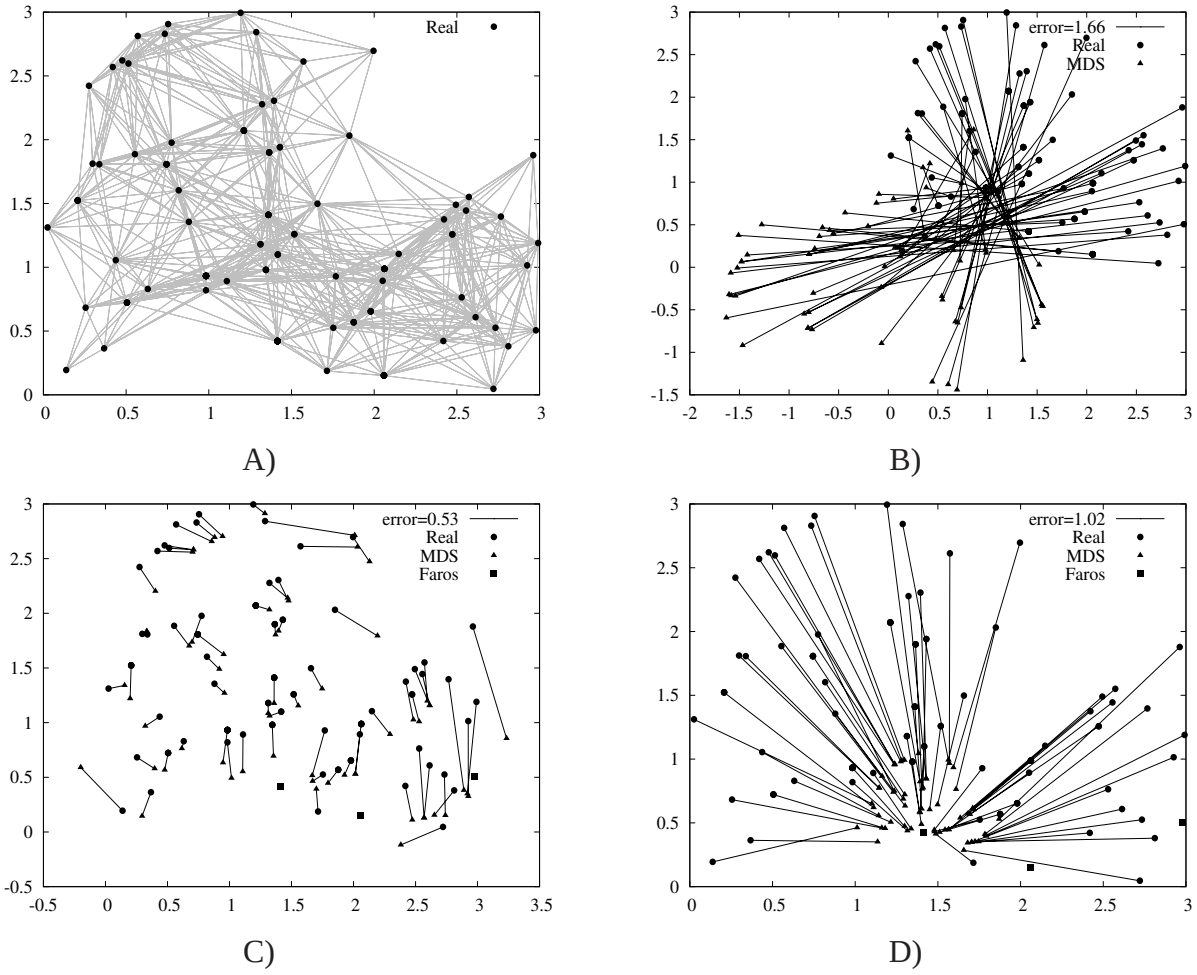


Figura 5.7: A) Red inalámbrica de 70 sensores, B) error en la reconstrucción MDS únicamente con las posiciones relativas, C) error en la reconstrucción cuando se ajusta la solución MDS a la solución real mediante el uso de faros (salto=0.8R) y D) error en la reconstrucción MDS para la longitud del salto de 0.2R.

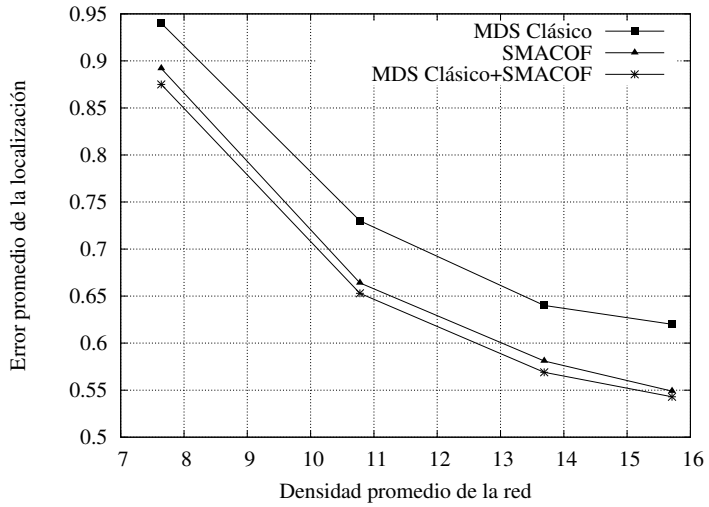


Figura 5.8: Influencia de la densidad de la red en el error de la localización para los tres algoritmos.

Se realizaron tres experimentos, el primero corresponde al análisis del algoritmo MDS Clásico (sección 3.3.6); el segundo experimento es para el análisis del algoritmo SMACOF (sección 3.3.8); finalmente, el tercero corresponde a una combinación de ambos, en la cual, se ejecuta primero el MDS Clásico y posteriormente el SMACOF para minimizar el error de la primera etapa. La necesidad de realizar estos tres experimentos no sólo es para estimar la longitud del salto que minimiza el error en la reconstrucción de la posición de cada algoritmo, sino también para comparar el error entre cada uno de estos algoritmos. En el Apéndice A se muestran las tablas de resultados; la Tabla A.2 muestra los de MDS Clásico, la Tabla A.3 los de SMACOF y la Tabla A.4 muestra el resultado de combinar ambos algoritmos.

Al igual que en el algoritmo Bellman-Ford, se puede observar un comportamiento común en los tres algoritmos que resuelven el MDS: el error en la reconstrucción MDS es muy grande cuando la red es rala y disminuye conforme aumenta la densidad. Este comportamiento se debe, principalmente, a la falta de rigidez de la gráfica, ya que en promedio una red rala no es tan rígida como una red densa [32]. Entonces, para que el MDS reconstruya las posiciones de los nodos de la manera más precisa requiere, preferentemente, que la conectividad de la gráfica sea lo mayor posible. En la Figura 5.8 se presenta la influencia de la densidad sobre los algoritmos MDS, nótese que las magnitudes del error son mayores, comparados con el Bellman-Ford, esto se debe a los factores de reconstrucción MDS mencionados anteriormente. Otro resultado importante que se presenta en la figura es el desempeño de los algoritmos, en primera instancia, puede observarse que el error en el algoritmo MDS Clásico+SMACOF es menor que los otros dos algoritmos. Este resultado se obtiene gracias a la combinación de los algoritmos, es decir, resolver el MDS Clásico y, posteriormente, minimizar el error en la reconstrucción mediante SMACOF. No obstante, emplear este algoritmo implica un costo computacional muy alto ($O(i \times n^3) + O(j \times n^2)$). Por otro lado, SMACOF es una buena opción para resolver la localización puesto que la diferencia promedio de error comparado con MDS Clásico+SMACOF es alrededor de 1 %, y el costo computacional es $O(j \times n^2)$. Esto resulta una ventaja importante ya que el costo computacional del SMACOF es menor que el MDS

Clásico y MDS Clásico+SMACOF. Además, la implementación del SMACOF puede resultar más sencilla para una WSN.

El cambio de la densidad de la red no representa gran variación en la elección adecuada del salto que minimiza el error entre las posiciones reales y las estimadas. Este comportamiento también puede observarse en los resultados mostrados en las Tablas A.2, A.3 y A.4 puesto que el algoritmo Bellman-Ford es la herramienta principal de los algoritmos MDS. Sin embargo, los valores adecuados del salto de cada tabla varían con respecto a los resultados del experimento Bellman-Ford. Esta variabilidad es normal debido a que la reconstrucción MDS incrementa el error, provocando que el valor adecuado del salto que minimiza el error se altere. ***Así, el valor del salto que ha de ser empleado por el algoritmo de localización es el que se presenta en estas tablas. Durante la ejecución del algoritmo cada líder decide qué longitud del salto emplear, dependiendo del tamaño de la colonia y del algoritmo seleccionado para la reconstrucción MDS***.

La Figura 5.9 es una comparación más de los algoritmos. No obstante, aquí se presenta el tamaño de la red contra el error en la localización para cuatro densidades diferentes. En primera instancia se observa que, al igual que en el resultado mostrado anteriormente (Figura 5.8), el algoritmo que en promedio reduce más el error de la localización es MDS Clásico+SMACOF, independientemente de la densidad.

Aunque en estas figuras se nota que el error aumenta cuando se incrementa el tamaño de la red, el error se atenúa conforme la densidad se incrementa. Por ejemplo, obsérvese que las pendientes que forman las curvas del SMACOF y el MDS Clásico+SMACOF en las Figuras 5.9 A), B), C) y D) cada vez se hacen más pequeñas.

Por otro lado, ***un resultado aún más importante es el siguiente: es evidente que el tamaño de la red también influye en el error, aunque no de manera considerable como lo hace la densidad. Ahora bien, estas curvas sugieren que se puede resolver la localización con el mínimo error posible siempre y cuando la red sea pequeña, independientemente de la densidad. Esto no sólo justifica que la partición de la red es necesaria para crear subredes más pequeñas, sino que también justifica el uso del algoritmo de particionamiento de Awerbuch, el cual permite parametrizar el criterio de crecimiento de las colonias variando su factor de crecimiento*** k***. Por tanto, es recomendable usar un valor para*** k ***grande para producir colonias más pequeñas.***

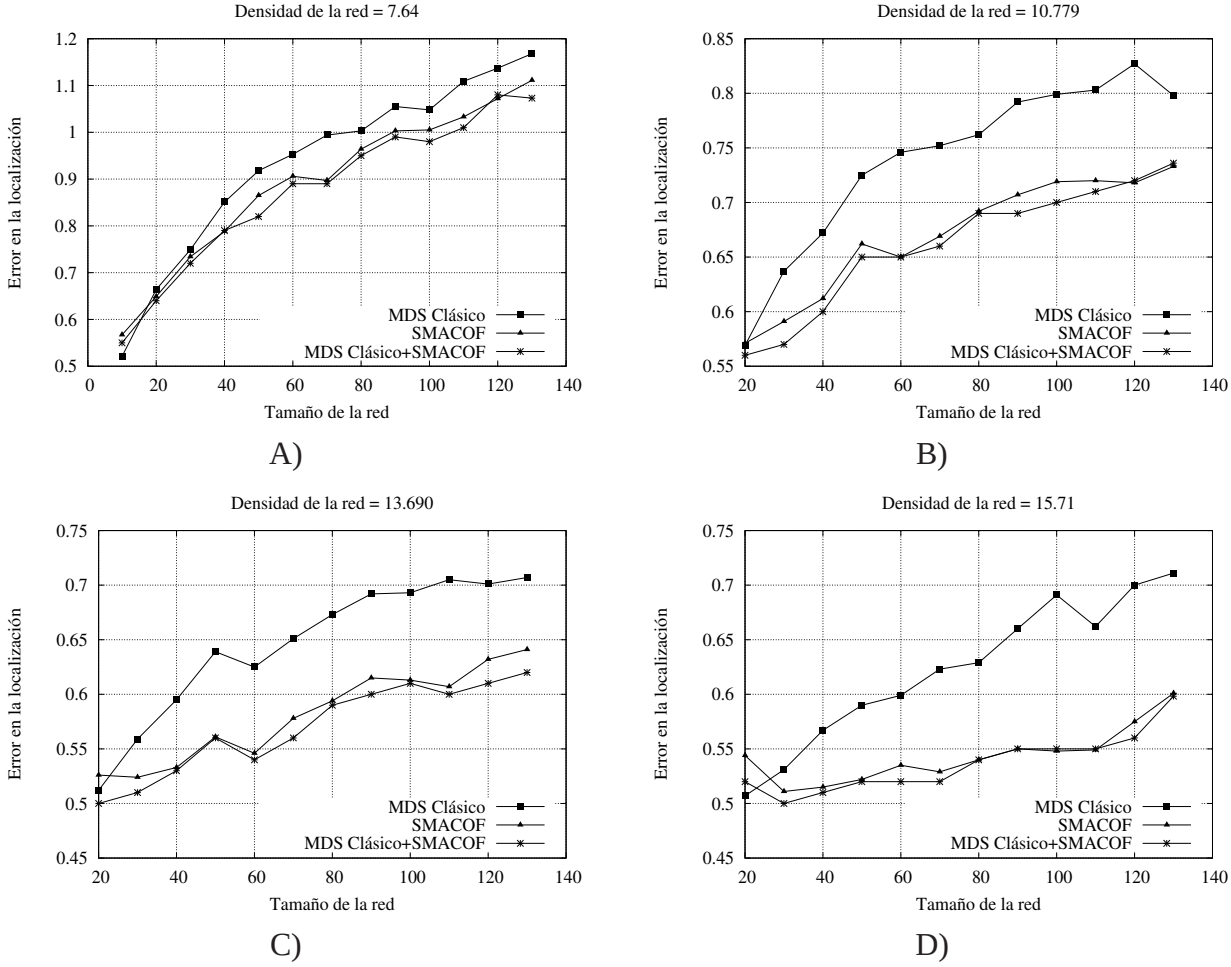


Figura 5.9: Comparación del error en la localización para los algoritmos que resuelven el MDS para redes de 10 a 130 nodos, con densidades de: A)7.64, B)10.78, C)13.690 y D)15.71$\frac{nodos}{R}$.

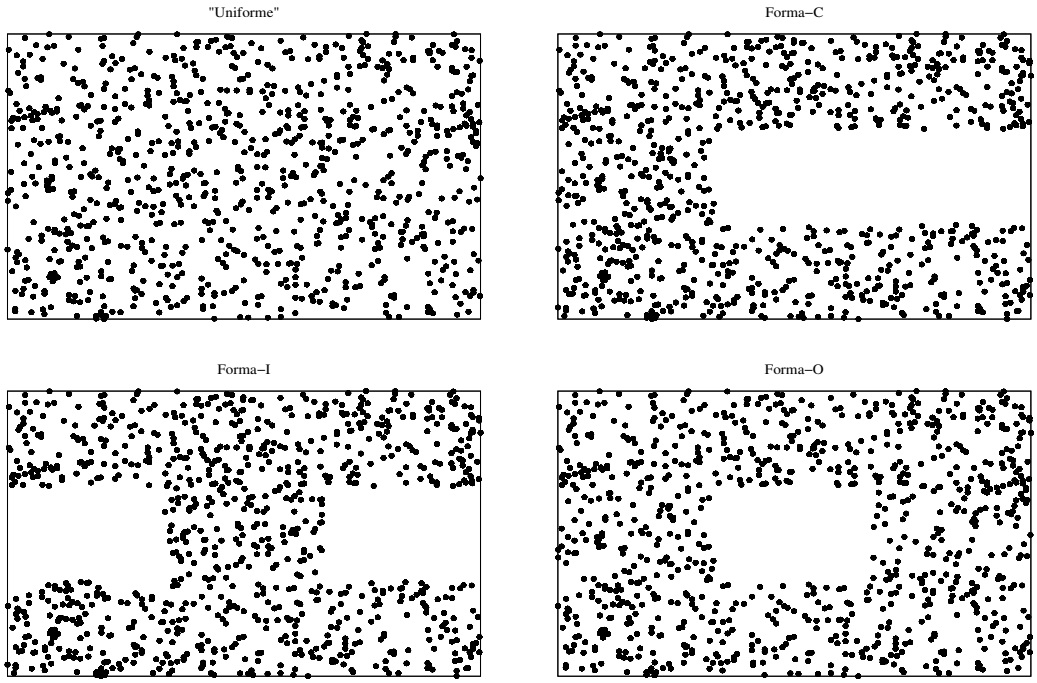


Figura 5.10: Cuatro topologías para evaluar el algoritmo de localización: red "*Uniforme* ", *Forma-C*, *Forma-I* y *Forma-O*

5.4. Localización

En esta sección se presentan los resultados del algoritmo de localización. Se aplicó el algoritmo correspondiente en cuatro redes con topología diferente: distribución *uniforme*, *forma-C*, *forma-I*, y finalmente, *forma-O* (Figura 5.10). Para evaluar el algoritmo de localización y comparar su desempeño contra el resultado de la localización cuando una red no se particiona, en cada una de las redes primero se resolvió la localización sin particionamiento y posteriormente particionando la red. Para medir el error en la localización cuando se resuelve en uno u otro método, se estimó el error cuadrático de las posiciones reales de los nodos con las estimadas con el algoritmo.

Por otro lado, en los resultados de la sección anterior se sugiere que el factor de crecimiento de las colonias k debe ser un valor grande para producir un mayor número de colonias pero con una cantidad de nodos menor y de esta manera, minimizar el error en la localización. Este efecto también se analiza aquí, se ejecutó el algoritmo de localización para valores de $k = 1$ y $k = 2$.

En las Figuras 5.11, 5.12, 5.13 y 5.14 se presenta el resultado de la localización para las redes con cuatro topologías diferentes: *uniforme, forma-I, forma-O* y *forma-C*, respectivamente. En estas figuras, los marcadores "•"representan las posiciones reales de los nodos mientras que "▲"las posiciones resultantes del algoritmo. Finalmente, "—"es el error en la distancia de la posición calculada a la posición real.

En cada una de las topologías se observan dos resultados importantes en común. Por un lado, el error de la localización de los nodos cuando la red no se particiona siempre es mayor, además, no

permite “reconstruir ”las topologías con obstáculos o irregularidades, esto no sólo es una desventaja en la calidad de la localización, sino también el costo computacional que ésta requiere. En segundo lugar, se observa que al particionar, independientemente del valor de k asignado, el error disminuye considerablemente debido a que las topologías irregulares se reconstruyen mejor.

Otro resultado importante relacionado con el particionamiento de la red, y que se muestra en los resultados de la sección anterior, es la influencia del factor de crecimiento de las colonias (k), donde el error es proporcional al tamaño de la red —*o colonia, en este caso*—. Por esto, en las figuras de los resultados se registra que para una $k = 2$ el error es menor, comparado con $k = 1$. Con esto se comprueba que el particionamiento en colonias más pequeñas permite localizar los sensores con mejores resultados puesto que las distancias de los nodos más cercanos tienen menos error que los más alejados, conjeturamos que desde el algoritmo Bellman-Ford se genera este comportamiento debido al cálculo de las distancias. Este efecto particular puede observarse mejor en la Figura 5.11 B), donde el nodo que inició el algoritmo de localización se encuentra aproximadamente en $(3, 6)$. Nótese que cuando los nodos se alejan cada vez más de esta coordenada, el error en su localización se incrementa.

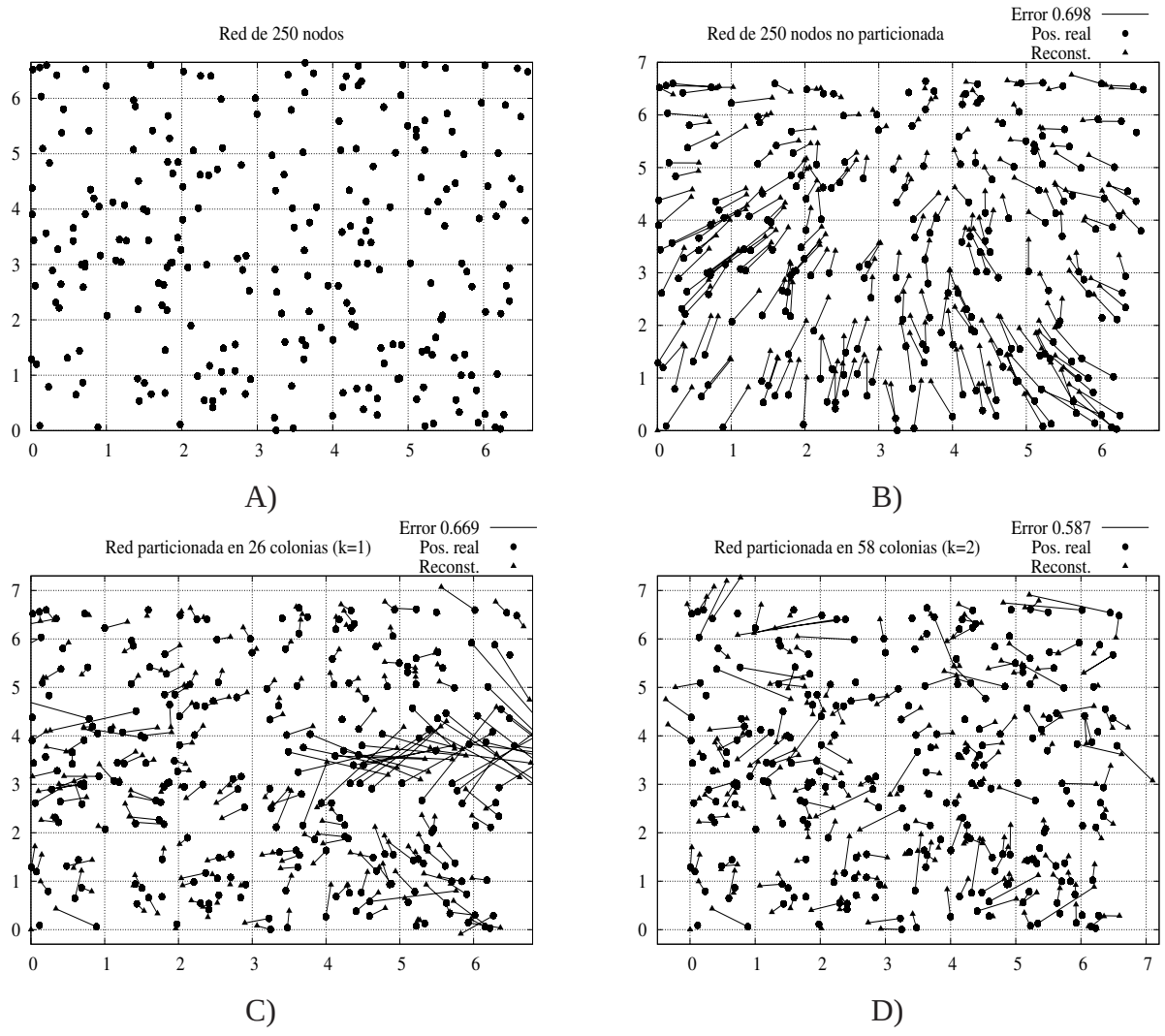


Figura 5.11: A) Red real de 250 nodos, B) resultado de la localización sin particionar la red, C) resultado de la localización para $k = 1$, D) y para $k = 2$.

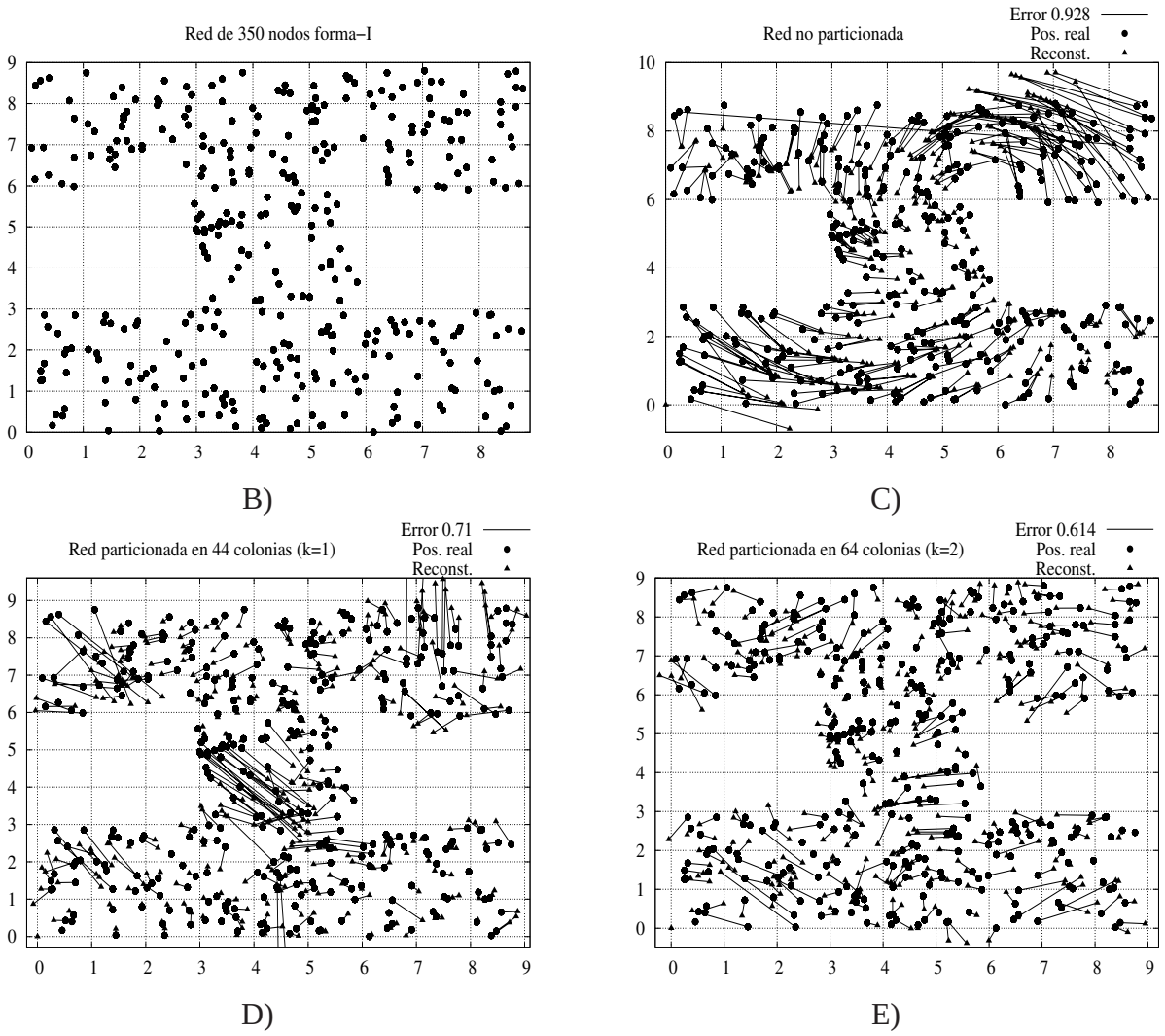


Figura 5.12: A) Red real de 350 nodos, B) resultado de la localización sin particionar la red, C) resultado de la localización para $k = 1$, D) y para $k = 2$.

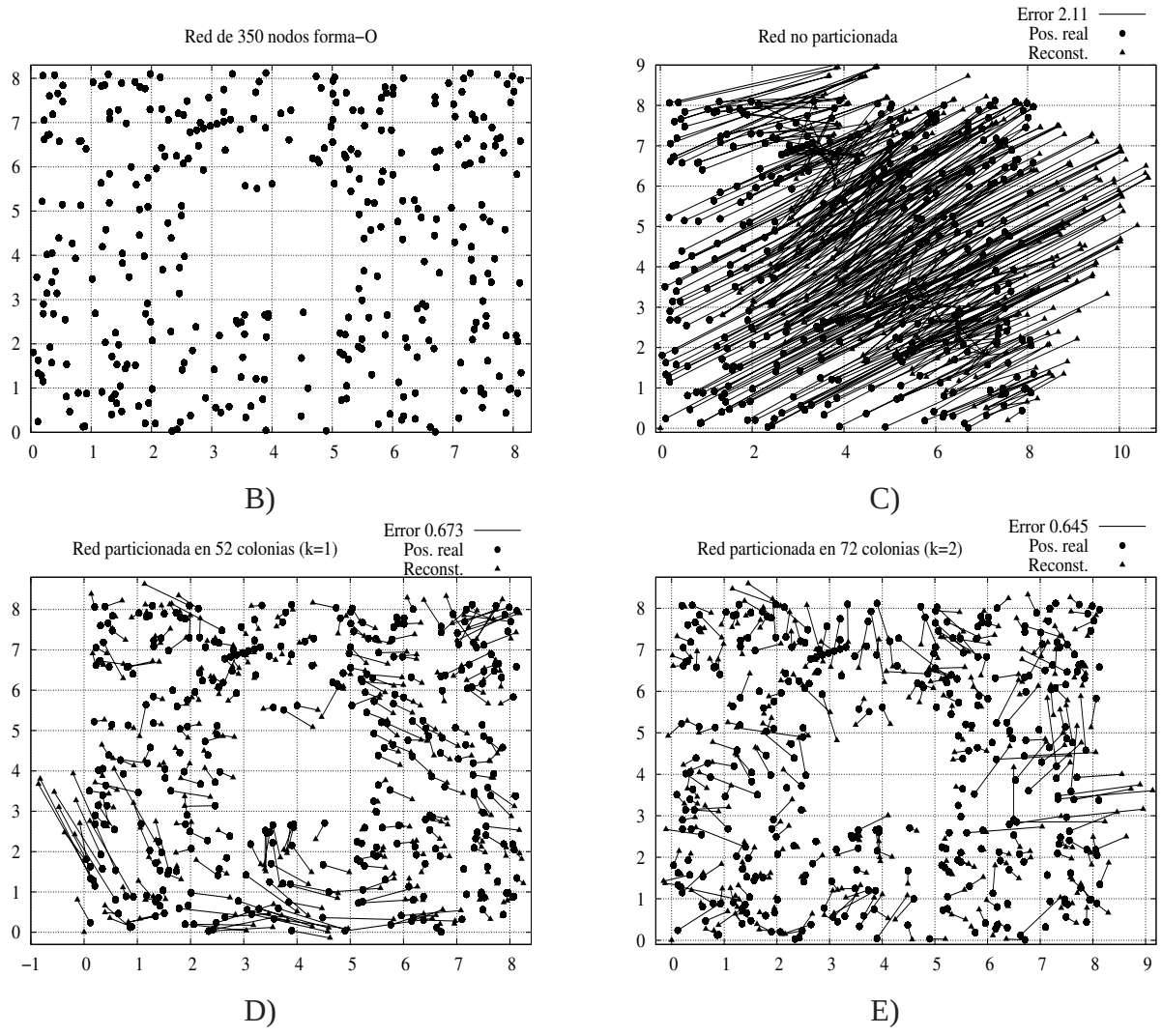


Figura 5.13: A) Red real de 350 nodos, B) resultado de la localización sin particionar la red, C) resultado de la localización para $k = 1$, D) y para $k = 2$.

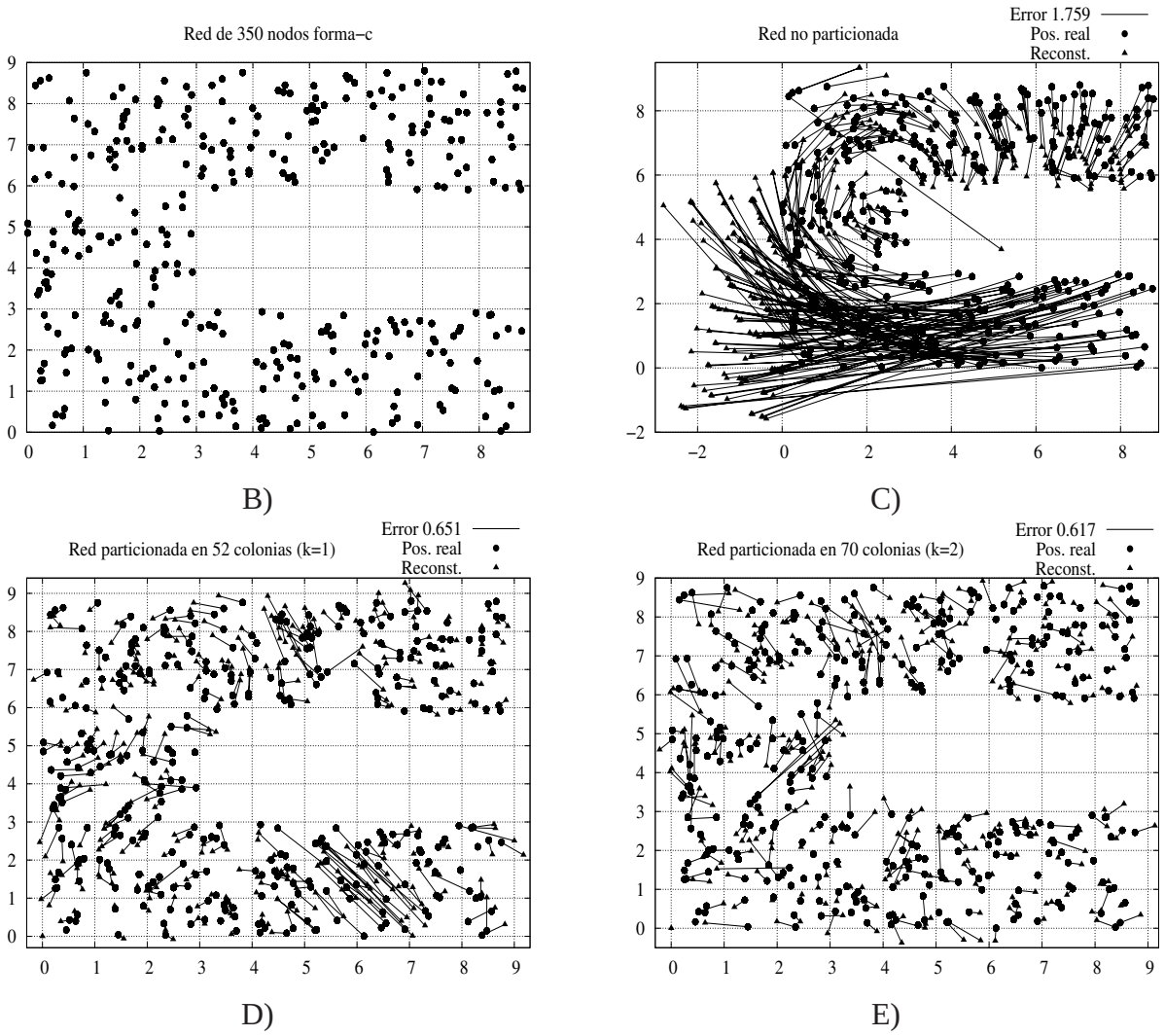


Figura 5.14: A) Red real de 350 nodos, B) resultado de la localización sin particionar la red, C) resultado de la localización para $k = 1$, D) y para $k = 2$.

Capítulo 6
Conclusiones y Trabajo Futuro

Particionamiento

El MDS es una herramienta muy aceptada para tratar el problema de la localización. Sin embargo, la cantidad de instrucciones requeridas es de orden $O(n^3)$ si es necesaria una eigendescomposición o $O(n^2)$ si se emplea SMACOF. Ahora bien, aunque se utilice un algoritmo MDS con complejidad modesta, como la de SMACOF, el proceso de localización requiere también un algoritmo de búsqueda de la mejor ruta cuya orden $O(|V|^3)$. Por esta razón, surge la necesidad de emplear un enfoque de *particionamiento* que permita disminuir aún mas los costos computacionales y de comunicaciones.

Este no es el primer trabajo que emplea un enfoque de partición para resolver el problema de la localización. La mayoría de las contribuciones particionan la red excesivamente de tal forma que cada nodo que subyace en la red es *líder* de colonia, el cual es el encargado de coordinar las operaciones en ella. Este comportamiento incrementa el costo computacional y de comunicaciones, puesto que cada nodo es miembro de varias colonias simultáneamente y por tanto, participa en distintas operaciones innecesariamente. En cambio, el algoritmo de particionamiento de Awerbuch aquí presentado, crea las colonias con un diámetro que puede parametrizarse para acotar los costos del computacionales y, en consecuencia, ningún nodo pertenece a varias colonias. Por otro lado, sólo el líder realiza cálculos en su colonia, esto permite reducir costos de ejecución del algoritmo.

Aquí se ha comprobado que siempre que una red pueda particionarse en varias colonias se obtiene un ahorro en operaciones y en mensajes. Por otro lado, el particionamiento también permite reconstruir las posiciones en redes con topologías irregulares y con obstáculos, esto se consigue gracias a que la división en distintas colonias y la ejecución independiente del MDS en cada una de ellas evaden las irregularidades.

En este trabajo también se presentó una variante del algoritmo de localización, el cual reduce el tiempo de finalización al igual que la cantidad de mensajes, este algoritmo busca *equilibrar* el número de mensajes intercambiados por los líderes de colonia y disminuye de manera razonable el tiempo de finalización del particionamiento debido a que emplea un enfoque de *paralelización*, esto es, cuando termina la creación de cierta colonia, inmediatamente comienza la formación de colonias contiguas de manera simultánea. En contraste, el algoritmo de Awerbuch crea las colonias *secuencialmente* mediante una búsqueda en profundidad. El cambio del criterio secuencial al paralelo no sólo implica que la búsqueda en profundidad se convierta en una búsqueda en amplitud, lo que reduce el tiempo de ejecución, sino que también disminuye de manera general la cantidad de

mensajes transmitidos por todos los nodos.

Una diferencia notable entre estos dos algoritmos es la distribución y tamaño de la partición resultante. Mientras que en Awerbuch las colonias son de tamaño similar, porque se construyen una tras otra, en el nuevo algoritmo cada colonia *contiende* con sus vecinas por los nodos que aún no se han integrado. Esta característica produce mayor variabilidad en el tamaño. Seleccionar el algoritmo modificado implica particionar toda la red en un tiempo relativamente bajo y con una modesta disminución de mensajes, a costa de la distribución y uniformidad del resultado. Sin embargo esto no es un factor importante en el desempeño del algoritmo de localización ya que, recordando, siempre que una red pueda particionarse los costos computacionales y de comunicaciones se reducen.

Los dos algoritmos de particionamiento pueden ofrecer cierta ventaja cuando se emplean en una WSN, esto es, al finalizar el particionamiento la red es estructurada como *bosque* donde cada uno de sus árboles es una colonia, esto implica que cada nodo dentro de la red conoce tanto a su ancestro inmediato como a sus descendientes y además, cada colonia es capaz de comunicarse con sus vecinas mediante la elección de un enlace único que las conecta. Esta estructura puede emplearse como una herramienta para asistir el encaminamiento [19].

Por otro lado, ***el tamaño de la red también influye en el error de la localización, aunque no de manera considerable como lo hace su densidad, esto sugiere que se puede resolver la localización con el mínimo error posible siempre y cuando la red sea pequeña, independientemente de la densidad. Esto no sólo justifica que la partición de la red es necesaria para crear subredes más pequeñas, sino que también justifica el uso del algoritmo de particionamiento de Awerbuch, el cual permite parametrizar el criterio de crecimiento de las colonias variando su factor de crecimiento*** k***. Por tanto, se concluye que usar un valor de*** k ***grande para producir colonias más pequeñas genera una mejor localización.***

Algoritmo Bellman-Ford

El algoritmo Bellman-Ford es una herramienta imprescindible para el MDS porque calcula la distancia en saltos (hops) entre cada pareja de nodos. Este algoritmo tiene una carga computacional de $O(|V|^3)$. De hecho, representa la etapa más costosa de la localización ya que en las etapas posteriores se utilizan algoritmos de menor complejidad.

Cabe mencionar que de las referencias consultadas sobre métodos basados en estimaciones libres de rango, ninguna propone asignar un valor del salto que minimice el error entre las distancias estimadas y las distancias reales. No obstante, ***empíricamente se encontró un valor del salto que minimiza el error en las distancias intra-nodos para redes de tamaño y densidad diferente. Esto no significa que el error sea pequeño, pero tampoco que el error es el mismo para las densidades consideradas (7.64, 10.78, 13.69 y 15.71*** $\frac{nodos}{R}$***). Así, observó que, independientemente del tamaño de la red, el error promedio en las distancias se incrementa cuando la densidad de la red es baja y que disminuye cuando la densidad aumenta. Este comportamiento implica que el error en la estimación de las distancias es inversamente proporcional a la densidad de la red.***

Estos resultados no pueden pasarse por alto ya que cada integrante de la red no es capaz de medir la distancia física con sus nodos vecinos. Así, el diseño del experimento permite observar que a pesar de esta carencia, el error en la estimación de las distancias entre cada par de nodos puede reducirse al emplear cierto valor para el salto, donde cada líder de colonia lo selecciona de

acuerdo con la densidad y el número nodos que yacen en su colonia.

Escalamiento Multi-Dimensional

El MDS debe resolverse con el error introducido en la matriz de distancias, pero también con otras fuentes de error involucrados en la reconstrucción, tal es el caso de la rigidez de la gráfica o de la hipótesis que asume la correlación entre la trayectoria más corta y la distancia euclidiana, la cual no siempre se cumple. Con respecto a la rigidez de la gráfica el MDS requiere, para una reconstrucción exacta, que los nodos involucrados estén conectados completamente, esto implica que la trayectoria más corta entre cada uno es, efectivamente, una línea recta. En un ambiente más restringido, como una WSN, resulta difícil que la gráfica sea completa, de hecho, por eso es necesario el algoritmo Bellman-Ford. Sin embargo, en aquellas redes con topologías irregulares, Bellman-Ford no es capaz de estimar con exactitud las distancias Euclidianas, independientemente de que las distancias físicas puedan medirse o no por cada uno de los nodos.

En este proyecto se propuso el uso de tres algoritmos que resuelven el MDS: MDS Clásico, SMACOF y por último la combinación de ambos. Cada uno de estos algoritmos cuenta con una complejidad de $O(i \times n^3)$, $O(j \times n^2)$ y $O(i \times n^3) + O(j \times n^2)$, respectivamente. Se evaluó el error en la reconstrucción de cada uno de ellos y se encontró que el error en la reconstrucción MDS es muy grande cuando la red es rala y disminuye conforme aumenta la densidad. Este comportamiento se debe, principalmente, a la falta de rigidez de la gráfica, ya que en promedio una red rala no es tan rígida como una red densa.

En cuanto al desempeño de los algoritmos se encontró, en primera instancia, que ***el error en el algoritmo MDS Clásico+SMACOF es menor que los otros dos algoritmos. Este resultado se obtiene gracias a la combinación de los algoritmos, es decir, resolver el MDS Clásico y, posteriormente, minimizar el error en la reconstrucción mediante SMACOF. No obstante, emplear este algoritmo implica un costo computacional muy alto. Sin embargo, SMACOF es una buena opción para resolver la localización puesto que la diferencia promedio de error comparado con MDS Clásico+SMACOF es alrededor de 1 %, y el costo computacional es mucho menor. Además, la implementación del SMACOF puede resultar más sencilla para una WSN, puesto que es un método iterativo.***

Mezcla de Colonias y Localización

El algoritmo de mezcla puede mejorarse si se disminuye la cantidad de faros que se necesitan ya que, recordando, se requieren al menos tres faros por colonia para completar la localización. Para superar esta restricción debe emplearse un nuevo enfoque de mezcla que requiera únicamente tres o cuatro faros para toda la red. Para esto, la mezcla de mapas no debe depender de la posición absoluta que obtiene mediante los faros, sino de la conectividad de las colonias, es decir, si cada colonia conoce los nodos que se encuentran en la periferia y además que comparten los límites con alguna otra, estos nodos pueden emplearse para *empalmar* las colonias, sin necesidad de los faros. Así los faros se emplean únicamente para construir un sistema absoluto de coordenadas sobre la red, compuesta por varias colonias.

A pesar de la exigencia de la cantidad de faros que requiere el algoritmo de localización

propuesto, se comprobó que las posiciones relativas de las colonias pueden reconstruirse con un error bajo, es decir, sin la participación previa de los faros.

Finalmente, cabe mencionar que emplear Awerbuch para resolver la localización en redes cuya topología es arbitraria, es comparable con los resultados presentados por Shang *et al*. [27]. Esto implica que emplear Awerbuch y ejecutar un sólo MDS sobre cada colonia no sólo reduce la complejidad del algoritmo de localización, sino que también produce una localización relativamente similar al referenciado algoritmo de Shang, en el cual, cada uno de los nodos ejecuta un MDS con sus vecinos más cercanos para formar las colonias, lo que provoca un incremento en la complejidad.

Trabajo Futuro

Con la propuesta presentada en este proyecto se lograron obtener resultados relativamente buenos, considerando que la única información de la que dispone cada nodo es únicamente de conectividad, esto es, *quién está en el radio de comunicaciones de quién*. Esta condición no afecta de manera considerable el MDS ya que, como se comprobó bajo experimentación, es posible que el líder seleccione una longitud del salto que minimice el error en la localización en cada colonia. Ahora bien, ¿es posible reducir aún más el error de la localización? Tal vez la respuesta pueda encontrarse si se cambia a un esquema basado en rango, es decir, que las distancias entre un par de nodos sean medidas. Con estas mediciones el MDS mejoraría la reconstrucción de la red, ya que las distancias empleadas se asemejarían a las distancias reales. El uso de este esquema implicaría cambios mínimos en el algoritmo de localización aquí propuesto y, además, la gran mayoría del hardware de radiofrecuencia, inclusive los dispositivos más económicos, tienen integrado el RSSI para medir distancias.

Si una WSN se despliega en una zona irregular con varios relieves a lo largo y ancho, entonces la reconstrucción de la red en dos dimensiones no es una opción. Como se ha mencionado a lo largo de este trabajo, el MDS es capaz de encontrar las posiciones en tres dimensiones. Es posible añadir esta característica al algoritmo de localización sin incrementar demasiado el costo computacional, de hecho, los límites superiores del consumo de recursos de tiempo y comunicaciones se mantienen tanto para el MDS Clásico como para el SMACOF. Esta adaptación requiere cambios mínimos únicamente en las etapas MDS y en las isometrías, además, como ya se ha mencionado el algoritmo de partición crea las colonias en cualquier dimensión.

Con respecto al algoritmo de mezcla, es necesario encontrar una metodología que mezcle las colonias con la información de conectividad entre ellas, esto para reducir la cantidad de faros requeridos en la localización. La idea de este nuevo algoritmo es que las colonias se mezclen utilizando la mayor cantidad de sensores que tienen en común, esto es, los que están en la periferia y son capaces de comunicarse con miembros de otra colonia. Mientras mayor sea este número, el posicionamiento relativo entre colonias se realiza con menor ambigüedad. Para terminar, se aplica a la mezcla la mejor isometría que transforma las coordenadas de una colonia a otra. Con esta metodología sólo se emplearían tres o cuatro faros para 2D y 3D, respectivamente.

Una característica más que puede añadirse al algoritmo de localización es la robustez. Hasta el momento, no se ha considerado la tolerancia a fallos en ninguna de las etapas del algoritmo, por ejemplo, si un nodo “muere ”durante el proceso de partición, esto podría detenerlo ya que este procedimiento se basa en un sistema de acuses para pasar a las etapas siguientes. Con respecto al

algoritmo Bellman-Ford, la situación es similar: la privación de un nodo también interrumpiría el algoritmo, puesto que la información se transmite hacia el líder por la estructura de árbol creada previamente, así, si todos los mensajes transmitidos al líder no llegan, entonces éste no creará la matriz de distancias y, por tanto, no se ejecutará el MDS.

En las tres etapas del algoritmo de localización: particionamiento, Bellman-Ford y MDS, es evidente que si un líder de colonia deja de trabajar provocaría que las etapas finalicen incorrectamente, o más aún, que no concluyan ya que los líderes, entre otras cosas, realizan todos los cálculos requeridos por la localización, por ejemplo, la eigendescomposición y estiman las mejores trayectorias entre cada pareja de nodos.

Apéndice A
Tablas de Resultados Experimentales

La Tabla A.1 muestra los resultados correspondientes del experimento Bellman-Ford. Para cada tamaño de red se generaron 8,000 redes aleatorias distribuidas en las cuatro densidades distintas dando como resultado 96,000 simulaciones en total. La primer columna de la tabla indica el tamaño de la red; la segunda, su densidad; la tercer columna muestra el mínimo error promedio que se consigue con la longitud del salto seleccionada; la cuarta columna es la desviación estándar correspondiente a ese valor promedio; finalmente, la quinta columna muestra la longitud del salto seleccionada que minimiza el error.

Las Tablas A.2, A.3 y A.4 muestran los resultados experimentales de la longitud del salto que minimiza el error en la localización usando los tres algoritmos que resuelven el MDS: MDS Clásico, SMACOF y MDS Clásico+SMACOF, respectivamente. El experimento consiste en crear redes aleatoriamente de 10, 20,...,130 nodos con 4 distintas densidades: 4, 6, 8 y 10 $\frac{Nodos}{R}$. En cada red se ejecuta el algoritmo Bellman-Ford, variando la longitud del salto de $0.1R$, $0.2R$... hasta R para construir la matriz de distancias. Para cada salto, se ejecutan los algoritmos MDS con la asistencia de tres faros para ajustar la reconstrucción los más posible a la solución real. Una vez que se tiene este ajuste, se mide el error entre la reconstrucción MDS y las posiciones reales y se selecciona el valor del salto para el cual el error es mínimo.

Tabla A.1: Estimación del salto en Bellman-Ford: longitud del salto para distintas colonias y varias densidades.

Tamaño de la colonia [*nodos*]	Densidad $\left[\frac{nodos}{area}\right]$	Mínimo Error	Desv. Estándar	Longitud del salto
10	7.64	0.228	0.023	0.6
	-	-	-	-
	-	-	-	-
	-	-	-	-
	Promedio			**0.6**
20	7.64	0.300	0.130	0.6
	10.78	0.243	0.020	0.6
	13.69	0.228	0.012	0.6
	15.71	0.229	0.011	0.6
	Promedio			**0.6**
30	7.64	0.387	0.178	0.6
	10.78	0.281	0.055	0.7
	13.69	0.257	0.021	0.7
	15.71	0.251	0.013	0.6
	Promedio			**0.65**
40	7.64	0.458	0.221	0.6
	10.78	0.304	0.083	0.7
	13.69	0.258	0.022	0.7
	15.71	0.251	0.014	0.7
	Promedio			**0.675**
50	7.64	0.534	0.282	0.6
	10.78	0.315	0.090	0.7
	13.69	0.267	0.021	0.7
	15.71	0.255	0.025	0.7
	Promedio			**0.675**
60	7.64	0.643	0.383	0.6
	10.78	0.347	0.154	0.7
	13.69	0.278	0.034	0.7
	15.71	0.263	0.014	0.7
	Promedio			**0.675**
70	7.64	0.690	0.401	0.6
	10.78	0.356	0.139	0.7
	13.69	0.287	0.039	0.7
	15.71	0.270	0.018	0.7
	Promedio			**0.675**
80	7.64	0.780	0.499	0.6
	10.78	0.378	0.145	0.7
	13.69	0.300	0.070	0.7
	15.71	0.280	0.016	0.7
	Promedio			**0.675**
90	7.64	0.801	0.506	0.6
	10.78	0.415	0.244	0.7
	13.69	0.307	0.035	0.7
	15.71	0.291	0.051	0.7
	Promedio			**0.675**
100	7.64	0.867	0.557	0.6
	10.78	0.418	0.194	0.7
	13.69	0.316	0.060	0.7
	15.71	0.298	0.016	0.7
	Promedio			**0.675**
110	7.64	0.968	0.644	0.6
	10.78	0.436	0.239	0.7
	13.69	0.321	0.034	0.7
	15.71	0.306	0.015	0.7
	Promedio			**0.675**
120	7.64	0.992	0.627	0.6
	10.78	0.425	0.207	0.7
	13.69	0.331	0.033	0.7
	15.71	0.316	0.021	0.7
	Promedio			**0.675**
130	7.64	1.022	0.671	0.6
	10.78	0.438	0.224	0.7
	13.69	0.338	0.031	0.7
	15.71	0.325	0.017	0.7
	Promedio			**0.675**

Tabla A.2: Error en la reconstrucción con MDS Clásico (medida de error para distintas longitudes del salto)

Tamaño de la colonia [*nodos*]	Densidad $\left[\frac{nodos}{area}\right]$	Mínimo Error	Desv. Estándar	Longitud del salto
10	7.64	-	0.145	0.6
	-	-	-	-
	-	-	-	-
	-	-	-	-
	Promedio			**0.6**
20	7.64	0.663	0.230	0.6
	10.78	0.569	0.185	0.6
	13.69	0.512	0.183	0.7
	15.71	0.507	0.147	0.6
	Promedio			**0.625**
30	7.64	0.749	0.250	0.6
	10.78	0.637	0.254	0.6
	13.69	0.559	0.198	0.7
	15.71	0.530	0.194	0.7
	Promedio			**0.65**
40	7.64	0.851	0.305	0.6
	10.78	0.671	0.247	0.7
	13.69	0.595	0.232	0.7
	15.71	0.567	0.207	0.7
	Promedio			**0.675**
50	7.64	0.918	0.330	0.6
	10.78	0.730	0.287	0.7
	13.69	0.639	0.259	0.7
	15.71	0.590	0.231	0.7
	Promedio			**0.675**
60	7.64	0.953	0.340	0.6
	10.78	0.746	0.300	0.7
	13.69	0.625	0.236	0.7
	15.71	0.599	0.227	0.7
	Promedio			**0.675**
70	7.64	0.994	0.337	0.6
	10.78	0.751	0.299	0.7
	13.69	0.651	0.247	0.7
	15.71	0.622	0.247	0.7
	Promedio			**0.675**
80	7.64	1.002	0.341	0.6
	10.78	0.761	0.295	0.7
	13.69	0.673	0.252	0.7
	15.71	0.629	0.235	0.7
	Promedio			**0.675**
90	7.64	1.055	0.378	0.6
	10.78	0.792	0.310	0.7
	13.69	0.692	0.284	0.7
	15.71	0.660	0.266	0.7
	Promedio			**0.675**
100	7.64	1.048	0.395	0.7
	10.78	0.799	0.305	0.7
	13.69	0.693	0.266	0.7
	15.71	0.691	0.317	0.8
	Promedio			**0.725**
110	7.64	1.109	0.398	0.6
	10.78	0.802	0.312	0.7
	13.69	0.705	0.289	0.7
	15.71	0.662	0.257	0.8
	Promedio			**0.7**
120	7.64	1.138	0.396	0.6
	10.78	0.827	0.342	0.7
	13.69	0.701	0.248	0.7
	15.71	0.700	0.306	0.7
	Promedio			**0.675**
130	7.64	1.167	0.422	0.6
	10.78	0.798	0.335	0.7
	13.69	0.707	0.249	0.7
	15.71	0.711	0.326	0.8
	Promedio			**0.7**

Tabla A.3: Error en la reconstrucción con SMACOF (medida de error para distintas longitudes del salto)

Tamaño de la colonia [*nodos*]	Densidad $\left[\frac{nodos}{area}\right]$	Mínimo Error	Desv. Estándar	Longitud del salto
10	7.64	0.567	0.124	0.5
	-	-	-	-
	-	-	-	-
	-	-	-	-
	Promedio			**0.5**
20	7.64	0.650	0.230	0.6
	10.78	0.571	0.203	0.6
	13.69	0.526	0.173	0.6
	15.71	0.544	0.162	0.6
	Promedio			**0.6**
30	7.64	0.734	0.273	0.6
	10.78	0.591	0.234	0.7
	13.69	0.524	0.214	0.7
	15.71	0.511	0.219	0.7
	Promedio			**0.675**
40	7.64	0.788	0.303	0.6
	10.78	0.612	0.254	0.7
	13.69	0.533	0.209	0.7
	15.71	0.515	0.222	0.7
	Promedio			**0.675**
50	7.64	0.865	0.313	0.6
	10.78	0.661	0.289	0.7
	13.69	0.561	0.252	0.7
	15.71	0.522	0.241	0.7
	Promedio			**0.675**
60	7.64	0.906	0.336	0.6
	10.78	0.650	0.270	0.7
	13.69	0.546	0.226	0.7
	15.71	0.535	0.232	0.7
	Promedio			**0.675**
70	7.64	0.897	0.327	0.6
	10.78	0.669	0.291	0.7
	13.69	0.578	0.259	0.7
	15.71	0.529	0.243	0.7
	Promedio			**0.675**
80	7.64	0.964	0.365	0.6
	10.78	0.692	0.302	0.7
	13.69	0.594	0.278	0.7
	15.71	0.540	0.241	0.7
	Promedio			**0.675**
90	7.64	1.003	0.393	0.6
	10.78	0.707	0.324	0.7
	13.69	0.615	0.296	0.7
	15.71	0.550	0.234	0.7
	Promedio			**0.675**
100	7.64	1.005	0.369	0.6
	10.78	0.719	0.322	0.7
	13.69	0.612	0.265	0.7
	15.71	0.548	0.215	0.7
	Promedio			**0.675**
110	7.64	1.032	0.388	0.6
	10.78	0.720	0.296	0.7
	13.69	0.607	0.262	0.7
	15.71	0.549	0.179	0.7
	Promedio			**0.675**
120	7.64	1.072	0.416	0.6
	10.78	0.718	0.308	0.7
	13.69	0.632	0.292	0.7
	15.71	0.608	0.279	0.7
	Promedio			**0.675**
130	7.64	1.110	0.426	0.6
	10.78	0.733	0.321	0.7
	13.69	0.641	0.290	0.7
	15.71	0.601	0.259	0.7
	Promedio			**0.675**

Tabla A.4: Error en la reconstrucción con MDS Clásico+SMACOF (medida de error para distintas longitudes del salto)

Tamaño de la colonia [*nodos*]	Densidad $\left[\frac{nodos}{area}\right]$	Mínimo Error	Desv. Estándar	Longitud del salto
10	7.64	0.549	0.120	0.5
	-	-	-	-
	-	-	-	-
	-	-	-	-
	Promedio			**0.35**
20	7.64	0.638	0.234	0.6
	10.78	0.556	0.203	0.6
	13.69	0.500	0.158	0.6
	15.71	0.525	0.157	0.6
	Promedio			**0.6**
30	7.64	0.719	0.260	0.6
	10.78	0.569	0.232	0.7
	13.69	0.505	0.198	0.7
	15.71	0.503	0.218	0.7
	Promedio			**0.675**
40	7.64	0.786	0.311	0.6
	10.78	0.604	0.249	0.7
	13.69	0.525	0.208	0.7
	15.71	0.508	0.217	0.7
	Promedio			**0.675**
50	7.64	0.820	0.289	0.6
	10.78	0.650	0.290	0.7
	13.69	0.557	0.250	0.7
	15.71	0.556	0.238	0.7
	Promedio			**0.675**
60	7.64	0.885	0.332	0.6
	10.78	0.645	0.270	0.7
	13.69	0.539	0.219	0.7
	15.71	0.518	0.205	0.7
	Promedio			**0.7**
70	7.64	0.889	0.332	0.6
	10.78	0.658	0.283	0.7
	13.69	0.555	0.237	0.7
	15.71	0.520	0.232	0.7
	Promedio			**0.7**

Tamaño de la colonia [*nodos*]	Densidad $\left[\frac{nodos}{area}\right]$	Mínimo Error	Desv. Estándar	Longitud del salto
70	7.64	0.889	0.332	0.6
	10.78	0.658	0.283	0.7
	13.69	0.555	0.237	0.7
	15.71	0.520	0.232	0.7
	Promedio			**0.7**
80	7.64	0.952	0.362	0.6
	10.78	0.689	0.300	0.7
	13.69	0.591	0.274	0.7
	15.71	0.537	0.236	0.7
	Promedio			**0.675**
90	7.64	0.992	0.387	0.6
	10.78	0.688	0.297	0.7
	13.69	0.604	0.274	0.7
	15.71	0.548	0.236	0.7
	Promedio			**0.675**
100	7.64	0.981	0.349	0.6
	10.78	0.700	0.302	0.7
	13.69	0.606	0.264	0.7
	15.71	0.554	0.223	0.7
	Promedio			**0.675**
110	7.64	1.008	0.376	0.6
	10.78	0.706	0.274	0.7
	13.69	0.602	0.260	0.7
	15.71	0.551	0.192	0.7
	Promedio			**0.7**
120	7.64	1.078	0.417	0.6
	10.78	0.715	0.311	0.7
	13.69	0.610	0.258	0.7
	15.71	0.584	0.236	0.7
	Promedio			**0.675**
130	7.64	1.073	0.402	0.6
	10.78	0.736	0.321	0.7
	13.69	0.620	0.266	0.7
	15.71	0.608	0.265	0.7
	Promedio			**0.675**

Apéndice B

Iteraciones Requeridas para Resolver los Algoritmos MDS Clásico y SMACOF

En las Sección 3.3 se presentan tres algoritmos que encuentran las posiciones relativas de los nodos: MDS Clásico, SMACOF y MDS Clásico+SMACOF, este último una combinación de los primeros. Las complejidades son $O(i \times n^3)$, $O(j \times n^2)$ y $O(i \times n^3) + O(j \times n^2)$, respectivamente. Donde i y j son las iteraciones necesarias para que cada algoritmo encuentre las posiciones relativas de los nodos.

Ahora bien, las iteraciones i y j no pueden determinarse previamente. Por ejemplo, para el SMACOF la cantidad de iteraciones j depende de la configuración inicial de las posiciones al igual que el criterio de paro, es decir, la calidad de la solución.

Para observar el comportamiento de estas variables se generaron 2,000 redes aleatorias de 10 hasta 130 nodos. En cada red se ejecutaron los dos algoritmos (MDS Clásico y SMACOF) y se registró la cantidad de iteraciones promedio requeridas para obtener un error de 0.0001 en las soluciones finales, este resultado se muestra en la Figura B.1 A) donde, en primer lugar, se observa que para los dos algoritmos, la cantidad de iteraciones es función del tamaño de la red. En segundo, las iteraciones requeridas para completar el MDS Clásico son considerablemente más bajas que el SMACOF, sin embargo, este resultado no representa que el MDS Clásico sea más eficiente que el SMACOF, de hecho, debido a que la mayor carga computacional del MDS Clásico requiere n^3 instrucciones, este proceso debe realizarse i veces. En contraste, el SMACOF requiere sólo n^2 que se repiten j veces. En la Figura B.1 B) se presentan las instrucciones totales requeridas para finalizar los algoritmos y se observa que, independientemente de que el MDS Clásico requiera menos iteraciones, SMACOF resulta más eficiente.

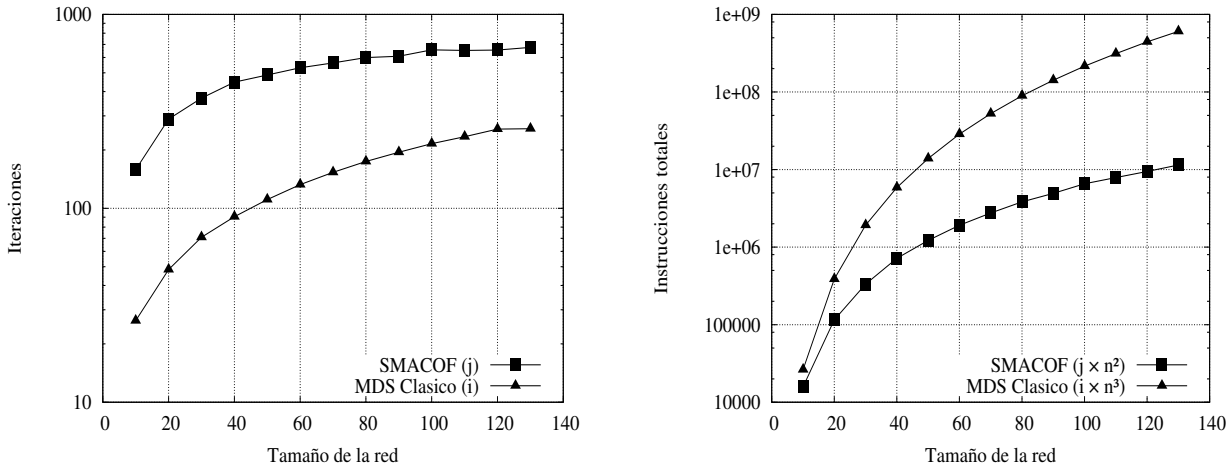


Figura B.1: A) Iteraciones requeridas para completar el MDS Clásico y SMACOF, B) instrucciones totales requeridas por ambos algoritmos.

Referencias

[1] H. Akcan, H. B. Vassil Kriakov, and A. Delis. Gps-free localization in mobile wireless sensor networks. Proceedings of the 5th ACM International Workshop on Data engineering for Wireless and Mobile Access, pages 35–42, June 2006.

[2] I. Akyildiz, W. Su, and Sankarasubramaniam. A survey on sensor networks. IEEE Communications MAgazine, 12(2):291–301, June 1991.

[3] B. Awerbuch. Complexity of network synchronization. Journal of the ACM (JACM), 32(4):745–770, October 1985.

[4] T. Banchoff and J. Wermer. Lineal Algebra Throught Geometry. Springer-Verlag, New York, 1992.

[5] R. Bellman. On a routing problem. The Quarterly of Applied Mathematics, 16(1):87–90, January 1958.

[6] R. Benítez. Geometria Vectorial. Trillas, México, 2002.

[7] J. A. Costa, N. Patwari, and A. H. III. Distributed weighted-multidimensional scaling for node localization in sensor networks. ACM Transactions on Sensor Networks (TOSN), 2(1):39–64, February 2006.

[8] T. Cox and M. Cox. Monographs on Statistics and Applied Probability 59: Multidimensional Scaling. Chapman and Hall, London, 1994.

[9] L. M. P. de Brito and L. M. R. Peralta. Collaborative localization in wireless sensor networks. In Proceedings of the 2007 International Conference on Sensor Technologies and Applications, pages 94–100. IEEE Computer Society, 2007.

[10] E. W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1:269–271, June 1959.

[11] L. R. Ford and D. Fulkerson. Flows in Networks. Princeton University Press, 1962.

[12] GNU-Octave. http://www.gnu.org/software/octave/. November 2009.

[13] P. Groenen and I. Borg. Modern Multidimensional Scaling, Theory and Applications. Springer-Verlag, New York, 1997.

[14] G. Kannan, G. Mao, and B. Vucetic. Simulated annealing based wireless sensor networks. In Vehicular Technology Conference, VTC 2006-Spring. IEEE 63rd, pages 1022–1026, May 2006.

[15] H. Karl and A. Willig. Protocols and Architectures for Wireless Sensor Networks. John Wiley & Sons, England, 2005.

[16] J. Kurose and K. Ross. Computer Networking. A Top-Down Approach Featuring the Internet. Pearson-Addison-Wesley, 2005.

[17] N. Mahalik. Sensor Networks Configuration: Fundamentals, Standards, Plataforms and Applications. Springer-Verlag, New York, 2007.

[18] G. Mao and B. Fidan. Localization Algorithms and Strategies for Wirless Sensor Networks. Information Science Reference, New York, 2009.

[19] R. Marcelín-Jiménez. Locally-constructed trees for ad-hoc routing. Telecommunication Systems, 36(1-3):39–48, January 2007.

[20] R. Marcelín-Jiménez, R. Esquivel-Villafaña, and S. Rajsbaum. A flexible simulator for distributed algorithms. Mexican International Conference on Computer Science, 0(0):39–48, September 2003.

[21] R. Marcelín-Jiménez, M. Ruiz-Sánchez, M. López-Villaseñor, V. Ramos-Ramos, C. Moreno-Escobar, and M. Ruiz-Sandoval. A survey on Localization in Wireless Sensor Networks. IGI Global, 2010.

[22] D.Ñiculescu and B.Ñath. Dv based positioning in ad hoc networks. Telecommunication Systems, 22(1-4):267–280, January 2003.

[23] D. Peleg. Distributed Computing. A Locality-Sensitive Approach. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[24] M. A. Pollatschek. A library for discrete event simulations. ACM SIGSMALL/PC Notes, 19(1):87–90, June 1993. Available at ftp://ftp.technion.ac.il/pub/supported/ie/bani/sss.zip.

[25] V. Ramadurai and M. Sichitiu. Localization in wireless sensor networks: A probabilistic approach. Proc. of the 2003 International Conference on Wireless Networks (ICWN 2003), pages 275–281, June 2003.

[26] P. K. Sahoo, I.-S. Hwang, and S.-Y. Lin. A distributed localization scheme for wireless sensor networks. Proceedings of the International Conference On Mobile Technology, Applications, And Systems, September 2008.

[27] Y. Shang and W. Ruml. Improved mds-based localization. In IN PROCEEDINGS OF IEEE INFOCOM '04, HONG KONG, pages 2640–2651, 2004.

[28] Y. Shang, W. Ruml, and Y. Zhang. Localization from mere connectivity. In International Symposium on Mobile Ad Hoc Networking and Computing, pages 201–212. ACM, March 2003.

[29] J. Shu, R. Zhang, Z. W. Linlan Liu, and H. Zhou. Cluster-based three-dimensional localization algorithm for large scale wireless sensor networks. Journal of Computers, 4(7):585–592, July 2009.

[30] J. Shu, R. Zhang, L. Liu, Z. Wu, and Z. Zhou. Cluster-based three-dimensional localization algorithm for large scale wireless sensor networks. Journal of Computers, Academy Publisher, 4(7):585–592, July 2009.

[31] P. Vicaire and J. Stankovic. Elastic localization: Improvements on distributed, range free localization for wireless sensor networks. University of Virginia, Technical Report CS-2004-35, 2004.

[32] C.-H. Wua, W. Sheng, and Y. Zhang. Mobile sensor networks self localization based on multi-dimensional scaling. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 4038–4043. IEEE, May 2007.

Índice Alfabético

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00001
Matrícula: 208382775

LOCALIZACION EN REDES INALAMBRICAS DE SENSORES

En México, D.F., se presentaron a las 12:00 horas del día 21 del mes de enero del año 2011 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. JESUS GUILLERMO FIGUEROA NAZUNO
DR. MICHAEL PASCOE CHALKE
DR. ENRIQUE RODRIGUEZ DE LA COLINA
DR. RICARDO MARCELIN JIMENEZ

Bajo la Presidencia del primero y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRO EN CIENCIAS Y TECNOLOGIAS DE LA INFORMACION

DE: CARLOS ERNESTO MORENO ESCOBAR

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, el presidente del jurado comunicó al interesado el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

CARLOS ERNESTO MORENO ESCOBAR
ALUMNO

REVISÓ

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE ANTONIO DE LOS REYES HEREDIA

PRESIDENTE

DR. JESUS GUILLERMO FIGUEROA NAZUNO

VOCAL

DR. MICHAEL PASCOE CHALKE

VOCAL

DR. ENRIQUE RODRIGUEZ DE LA COLINA

SECRETARIO

DR. RICARDO MARCELIN JIMENEZ