Comparación de un método exacto y aproximado en la resolución del TSP para una WSN

Jonathan Eduardo Tito Ontaneda, Marco Esteban Yacelga Pinto
Escuela Politécnica Nacional (EPN) Quito - Ecuador

Resumen **– El presente artículo describe el problema del agente viajero (TSP — Traveling Salesman Problem) en una red inalámbrica de sensores (WSN — Wireless Sensor Network) simulada a través de Castalia. En particular, los métodos de resolución del TSP, por árbol de expansión mínima (MST — Minimum Spanning Tree) con el algoritmo 2-opt y por el método de ramificación y poda (B&B — Branch and Bound). Así mismo, se resuelve el problema MST en el menor tiempo, a través de la simulación, la cual define dos escenarios de despliegue para tres modelos de motas: TelosB, Imote2 y Zolertia. Finalmente, se compara parámetros, tales como: throughput y consumo de energía para todas las combinaciones de escenario, modelo de nodo y método de resolución del TSP, concluyendo así cuál es el mejor método aplicable a una WSN.**

Índices **–** ***WSN, Sistemas Embebidos, Comunicaciones inalámbricas, Internet de las Cosas, Travelling Salesman Problem, Teoría de Grafos.***

I. Introducción

Las WSN se han vuelto ubicuas y son utilizadas en un amplio rango de dominios de aplicación [1], donde su importancia se refleja en su gran número de aplicaciones dada la gran cantidad de parámetros medibles por los sensores. Por otro lado, el TSP es quizás el problema de optimización combinatoria más estudiado a lo largo de la historia, el cual consiste que un agente viajero debe recorrer todas las ciudades y cada una de ellas una sola vez hasta llegar a la ciudad inicial de su periplo. Sin embargo, existe el condicionante de que debe utilizar el camino más corto posible, donde se pueden tener varios agentes viajeros o se pueden considerar diferentes métricas. Consecuentemente, se produce una multitud de variantes y extensiones con infinidad de aplicaciones prácticas en la vida real. En 1972 Richard M. Karp demuestra que el TSP es del tipo NP-duro [2], el cual basta con resolver el TSP manualmente con un número inicial de 3 ciudades e ir aumentando el número de ciudades para constatar como la complejidad crece de forma exponencial.

El problema del TSP es adaptable a las redes de sensores inalámbricos (WSN), donde los parámetros medidos por los nodos deben ser recolectados y transportados dependiendo de la aplicación en cuestión. Según [3] es posible reducir el consumo de potencia de los nodos sensores y, por lo tanto, extender el tiempo de vida de la WSN, al reducir el número y tamaño de los datos comunicados.

Establecer mínimo costo para el intercambio de datos en una WSN, reduce el número y tamaño de los datos comunicados. Uno de estos esquemas para encontrar un camino de mínimo costo se lo define con la resolución del TSP, el cual podría aplicarse en la difusión eficiente de información en una red de datos. En la sección 2 se estudian dos métodos de resolución del TSP desde la perspectiva de la teoría de grafos. El primero se trata de un método heurístico (aproximado) que consiste en calcular el MST y tomar el costo total de recorrerlo desde y hasta el nodo fuente. Como el límite inferior para encontrar la solución óptima, se parte de la disposición de los nodos obtenida con el MST y mediante el algoritmo 2-opt se procede a transformar caminos cruzados hasta formar el tour de mínimo costo que resolvería el TSP. El segundo es un método exacto que se basa en la búsqueda (branch - ramificación) y descarte (bound - poda) de soluciones factibles del TSP. El criterio de discriminación se basa en la comparación del costo de las soluciones parciales con el límite para el tour óptimo que se obtiene de las ecuaciones e inecuaciones de Held-Karp.

Además, en la sección 3 se explica brevemente la abstracción mediante la cual se llega a definir la simulación de la WSN en cuestión para su posterior evaluación respecto a la resolución del TSP mediante los dos métodos descritos en la sección 2.

II. El TSP desde la perspectiva de la teoría de grafos

Existen algoritmos que han originado protocolos que poseen un origen matemático basado en la teoría de grafos. Por ejemplo, los algoritmos de Dijikstra o de Bellaman-Forman son parte esencial de protocolos de enrutamiento como OSPF o RIP respectivamente [4]. Esto se debe a que los flujos de información pueden ser modelados matemáticamente desde la perspectiva de los grafos, de tal manera que mediante los cálculos pertinentes se puede establecer el camino que deben seguir los flujos de información para cumplir ciertas condiciones planteadas.

A. Definiciones de teoría de grafos utilizadas en el presente trabajo

Según Vitaly Voloshin [4] un grafo, explicado de forma sencilla, es un conjunto de puntos y líneas que conectan algunos pares de puntos. En teoría de grafos, los puntos son llamados nodos o vértices y las líneas son llamadas aristas (*edges*). En [5]

se afirma que por lo general los nodos o vértices son entes de procesamiento o estructuras que contienen algún tipo de información, mientras que las aristas en cambio representan conexiones o relaciones entre los nodos.

Un *dígrafo* o *grafo dirigido* es un grafo que utiliza flechas para conectar los nodos, debido a que las aristas tienen una dirección, no obstante un dígrafo es un *grafo no dirigido* en el cuál las aristas no tienen dirección y se utiliza líneas para la representación de las aristas en este tipo de grafos. Frecuentemente, las aristas tienen algún tipo de información asociada (distancia, costo, confiabilidad, etc.), y como resultado se genera lo que se conoce como un *grafo pesado* (Véase Fig. 1 b) [5].

Adicionalmente, un *camino* es un grafo en el cuál todos los vértices pueden ser ordenados (ordenados de izquierda a derecha), de tal manera que exista precisamente una arista conectando a cada dos vértices consecutivos y no exista otras aristas, en cambio, un *ciclo* es un camino que termina en el mismo nodo donde comenzó. Si el camino de este ciclo recorre todos los nodos del grafo se le llama *tour* [6].

Se dice que un grafo es *conexo* cuando se puede llegar desde cualquier nodo hasta cualquier otro nodo mediante un camino. Si el caso anterior no se cumple, entonces se concluye que el grafo no es conexo, pero puede dividirse en componentes conexas [6]. El concepto de grafo conexo suele aplicarse generalmente a grafos no dirigidos, si en estos grafos existe una arista para cada par de nodos entonces el grafo se denomina *completo* [6].

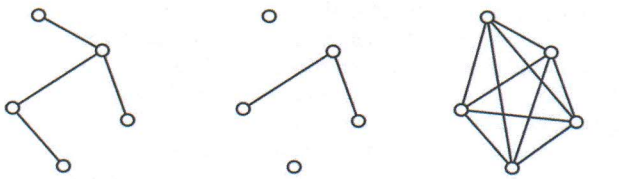


Fig. 1. a) grafo conexo, b) grafo no conexo y c) grafo completo

Además, dado un vértice, el número de todos los vértices adyacentes a él es llamado el *grado del vértice o nodo*, donde el máximo grado de entre todos los vértices del grafo es el *máximo grado del grafo*.

Por otro lado. El *camino de Hamilton* consiste en estudiar la existencia de un camino que conecte dos vértices dados de tal manera que visite cada nodo del grafo exactamente una vez, si el camino tiene como inicio y final el mismo vértice entonces se desea encontrar un *tour de Hamilton*, fácilmente se puede deducir que el máximo grado del grafo que conforma el tour siempre será igual a dos. Además, de todos los tours Hamiltonianos posibles para un mismo grafo, encontrar el tour Hamiltoniano de mínimo costo equivale a encontrar la solución al TSP.

B. Sobre la complejidad de los problemas de grafos

Un problema de procesamiento de grafos clasificado como *fácil* es aquel que se puede resolver utilizando un programa eficiente. Generalmente, se puede decir que el problema es fácil si podemos desarrollar algoritmos de fuerza bruta que nos permitan hallar la solución buscada en un tiempo de ejecución lineal (para el peor caso), o limitado por un polinomio de bajo grado en el número de nodos o el número de aristas.

Un problema *tratable* de procesamiento de grafos ocurre cuando se conoce un algoritmo que garantiza que sus requerimientos en tiempo y espacio están limitados por una función polinomial en el tamaño del grafo (número de nodos + número de aristas), por lo tanto, todo problema fácil es tratable.

Un problema *intratable* de procesamiento de grafos se caracteriza en que no se conoce algún algoritmo que garantice obtener una solución del problema en una cantidad razonable de tiempo, es decir, si se utilizase un método de fuerza bruta para probar todas las posibilidades para calcular la solución, existirían demasiadas posibilidades a considerar.

Existe una clasificación de tipos de problemas que se abordan dentro de la teoría de la complejidad, el primer grupo de problemas son aquellos agrupados en el conjunto "P", cuyos algoritmos de resolución son definidos por funciones polinomiales (el tiempo de ejecución está en función del número valores en la entrada), un segundo grupo es aquel denominado como "NP", problemas de decisión que no pueden resolverse en un tiempo polinómico.

El tercer grupo es denominado como NP-Completo, y son aquellos problemas NP más difíciles de resolver, y por último se encuentra el grupo de NP-Duros o problemas intratables, debido al espacio de soluciones que se tiene a medida que crece el número de instancias involucradas en cada uno de los problemas. En particular, la clase NP-Completo puede definirse alternativamente como la intersección entre NP y NP-Duro.

C. Definición del TSP

Encontrar un tour hamiltoniano para un grafo cualquiera es un problema intratable, esto se debe a que la única solución conocida para el caso general del problema consiste en buscar la solución entre todas las posibilidades de ciclos en el grafo. Como la gran mayoría de los problemas intratables, el problema de encontrar un tour hamiltoniano es un problema NP-duro, otros problemas que son variaciones del tour hamiltoniano, heredan el mismo nivel de dificultad, este es el caso del problema del agente viajero (TSP).

Su definición matemática la extraemos de [14], sea:

$$G = (X, E) \tag{1}$$

Donde G es un grafo (dirigido o no dirigido), X es el conjunto de vértices o nodos que conforman el grafo G, y E es el conjunto de aristas que conforman el grafo G. Para cada arista $e \in E$,existe un costo (peso) c_{e_i} que es prescrito. Entonces el TSP consiste en encontrar un tour Hamiltoniano en G de tal manera que la suma de los costos de las aristas del tour sea la más pequeña posible.

Sin perder generalidad, se puede asumir que G es un grafo completo (esta es una condición para que el TSP sea resoluble en G), de otra manera se pueden reemplazar las aristas faltantes con aristas de un costo muy grande, siendo:

$$E = \{e_1, e_2, \ldots e_m\} \tag{2}$$

Donde e_1, e_2, e_3 y e_m son las aristas pertenecientes al grafo G. Además, siendo matriz de costos, matriz de distancias, matriz de pesos o matriz de adyacencias, donde C = (C_{ij}) es la matriz de costos, entonces la entrada ij-ésima c_{ij} corresponde al costo de la arista que une el vértice i y el vértice j en G.

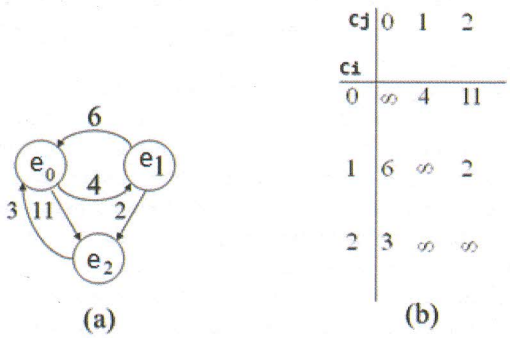


Fig. 2. (a) Digrafo G. (b) Matriz de costos para G, convirtiendo a G en un digrafo completo [15]

Además, dependiendo de la naturaleza de la matriz de costos (equivalentemente a la naturaleza de G), el TSP es dividido en dos clases. Si C es simétrica (esto es, G es no dirigido) entonces el TSP es llamado el problema del agente viajero simétrico (STSP – Symmetric Traveling Salesman Problem). Si C no es necesariamente simétrico (equivalentemente a que el grafo G es dirigido) entonces este es llamado el problema del agente viajero asimétrico (ATSP – Asymmetric Traveling Salesman Problem).

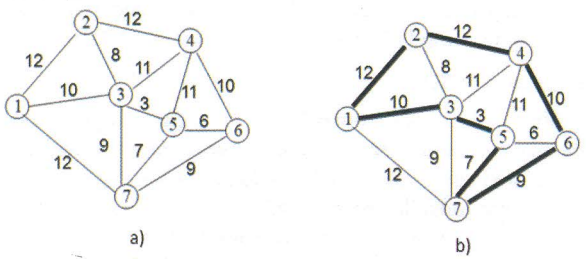


Fig. 3. En a) un grafo no dirigido y en b) su TSP resuelto resaltado en negrilla [15].

Como se puede apreciar en la Fig. 3, resolver el TSP para un número reducido de nodos puede realizarse sin dificultad manualmente evaluando todas las alternativas, pero cuando el problema crece en número de nodos es necesario acudir a uno de los métodos aproximados o a alguno de los métodos exactos para obtener una respuesta en una cantidad razonable de tiempo.

III. SIMULANDO EL TSP EN UNA WSN

A. Definiciones sobre redes inalámbricas de sensores

Una red inalámbrica de sensores según [5] es un sistema autónomo auto organizado de nodos sensores, distribuidos sobre una determinada área, para sensar datos de sus alrededores y agruparlos hacia un nodo fuente para su futuro procesamiento.

Para ampliar la definición anterior, podemos citar a [16] que define una WSN como una colección de pequeños dispositivos de alimentación autónoma (comúnmente llamados nodos sensores), dotados con capacidades de sensado, comunicación y procesamiento.

Una vez desplegados los nodos sensores pueden capturar datos de alguna magnitud física tal como la temperatura, presión atmosférica o alguna concentración de contaminantes. Las lecturas de los sensores son usualmente reportadas a un servidor central, también llamado nodo fuente, donde estas lecturas serán posteriormente procesadas de acuerdo a los requerimientos de la aplicación.

Adicionalmente, un nodo fuente puede ser cualquiera de una variedad de dispositivos, por ejemplo, una computadora portable, un PDA (Personal Digital Assistant – Ayudante Personal Digital), una estación terrena, etc.

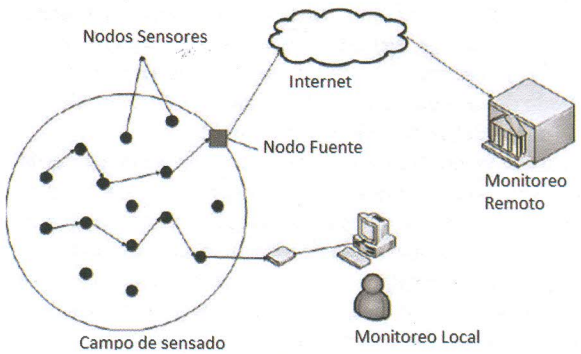


Fig. 4. Estructura de una WSN

Dependiendo de la aplicación de la WSN en cuestión, se puede tener más de un nodo fuente, en [17] existe un estudio sobre el tema de las WSN con múltiples nodos fuente.

Según [18] un nodo sensor está conformado por los subsistemas de sensado, procesamiento, comunicaciones y potencia. El diseñador tiene una plétora de opciones, en cuanto a decidir cómo construir y disponer juntos estos subsistemas unificados en un nodo programable.

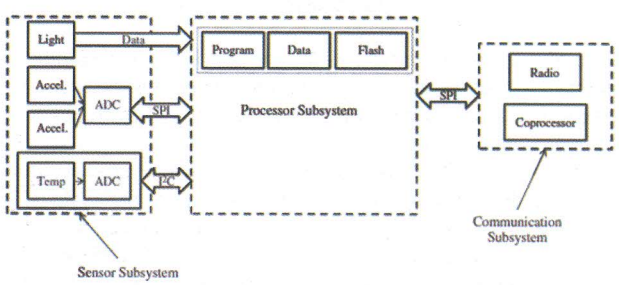


Fig. 5. Arquitectura de un nodo sensor inalámbrico

En la Figura 5, empezando desde la esquina superior izquierda se puede ver un sensor de luz denominado light (luz) que envía datos al subsistema de procesamiento. A continuación, se pueden observar dos sensores, uno de aceleración y uno de temperatura conectados a ADC (conversor analógico digital). Algunos sensores tienen su propio ADC incorporado, el cual puede ser directamente comunicado con el procesador. Los datos procesados son enviados inalámbricamente gracias al subsistema de comunicaciones.

La capa física direcciona las necesidades de simples, pero robustas técnicas de modulación, transmisión y recepción. Debido a que el entorno es ruidoso y que los nodos sensores pueden ser móviles. La capa de enlace de datos es la responsable de asegurar comunicaciones confiables a través de técnicas de control de errores y gestión de acceso al canal a través de MAC para minimizar la colisión por difusión con los vecinos. Dependiendo de las tareas de sensado, diferentes tipos de aplicación de software pueden ser construidos y usados en la capa de aplicación. La capa de red se encarga del enrutamiento de los datos que le provee la capa de transporte. La capa de transporte ayuda a mantener el flujo de datos si la aplicación de la WSN lo requiere así.

Además, los planos de potencia, movilidad y gestión de tareas monitorean la potencia, el movimiento y la distribución de tareas entre los nodos sensores. Estos planos ayudan a los nodos sensores a coordinar las tareas de sensado y minimizar el consumo promedio de energía. El plano de gestión de potencia su cometido es gestionar el consumo de energía del nodo. Por ejemplo, el nodo sensor puede apagar su receptor después de recibir un mensaje de uno de sus vecinos, para evitar tener mensajes duplicados. También, cuando el nivel de potencia de un nodo sensor es bajo, dicho sensor difunde a sus vecinos que su potencia es baja y que no puede participar en los mensajes de enrutamiento. De hecho, la potencia sobrante es reservada para el sensado. El plano de gestión de movilidad detecta y registra los movimientos de los nodos sensores, de tal manera que una ruta de regreso hacia el usuario es siempre mantenida, y los nodos sensores pueden mantener la pista de sus vecinos. Mediante el conocimiento de estos nodos sensores vecinos, los nodos sensores pueden balancear su potencia y uso de tareas. El plano de gestión de tareas balancea y planifica las tareas de sensado dadas a una región específica. No todos los nodos sensores en dicha región son requeridos para llevar a cabo tareas de sensado al mismo tiempo.

B. El TSP aplicado a las WSN

Al aplicarse el problema del agente viajero a una WSN, las ciudades se convierten en nodos sensores, el agente viajero pasa a ser un paquete de datos, la ciudad de origen del recorrido se torna en el nodo sumidero y los costos de las rutas a recorrerse para resolver el problema, podrían ser cuantificados al evaluar la calidad del enlace según los parámetros que la aplicación de la WSN considere críticos (consumo de energía, BER[1] , LQI[2] , etcétera).

Evaluar rutas eficientes, encasilla al TSP en la capa de red (enrutamiento), además el tiempo de vida útil para los nodos sensores aumenta si colaboran entre ellos compartiendo recursos de comunicación, por tal motivo el estudio de la resolución del TSP en una WSN está orientado a comunicar a todos los nodos que pertenezcan a la WSN de manera eficiente considerando el nodo fuente como el nodo principal.

Debido a que el TSP solo es resoluble en grafos completos, esto implica que la WSN debe tener la mayor conectividad posible de tal manera que se pueda cuantificar la gran mayoría de sus enlaces, caso contrario, tarde o temprano el TSP terminará tomando un enlace de costo infinito (un enlace inexistente en la WSN), esto debido a que se debe cumplir la condición de que el tour debe pasar por todos los nodos una vez, tal como se muestra en la Fig. 6.

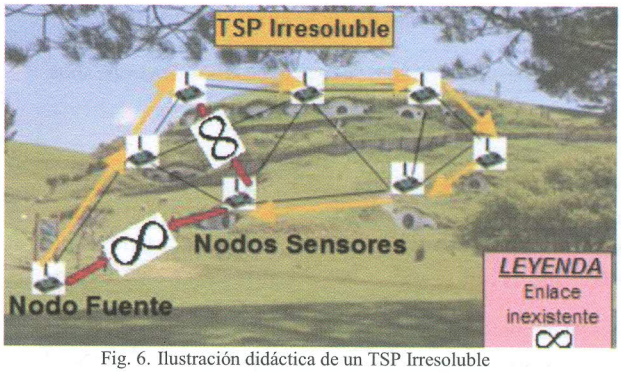


Fig. 6. Ilustración didáctica de un TSP Irresoluble

C. Simulador Castalia

Fue desarrollado por el NICTA (National Information Communication Technology Australia) para simular Redes Inalámbricas de Sensores (WSN), Redes de área corporal (BAN) y en general de redes de dispositivos embebidos de baja potencia. Es utilizado por investigadores y desarrolladores para probar sus algoritmos distribuidos y protocolos en modelos de canal y radio inalámbricos realistas, con un comportamiento de nodo objetivo especialmente relacionado en lo que respecta al acceso al medio [20]. Castalia es una herramienta de código abierto genérica, es decir en ella no se puede probar código o hardware para replicarlo exactamente en un modelo de nodo sensor específico [21].

Castalia simula todos los componentes de los nodos sensores reales, en la Fig. 9 las flechas de línea continua significan que existe un paso de mensajes, mientras las flechas punteadas significan que existe un simple llamado de funciones.

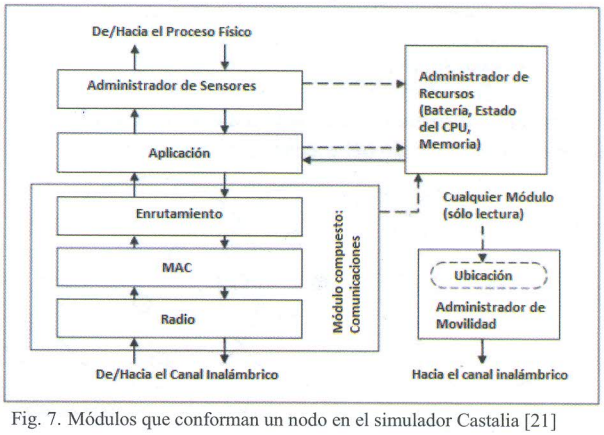


Fig. 7. Módulos que conforman un nodo en el simulador Castalia [21]

Módulo de Aplicación. - Es el módulo que comúnmente el usuario puede cambiar, usualmente creando un nuevo módulo para implementar un nuevo algoritmo.

Módulos de Comunicación: MAC y Enrutamiento. – Usualmente serán modificados por el usuario para implementar un nuevo protocolo.

[1] *Bit Error Ratio* – Tasa de bits errados

[2] *Link Quality Indicator* – Indicador de calidad del enlace

Módulo Administrador de movilidad. – El usuario generalmente modificará este módulo para crear un nuevo patrón de movilidad.

D. Modelos de mota utilizados en la simulación

TABLA I
RESUMEN DE LOS DATOS TÉCNICOS DE LOS MODELOS DE MOTA UTILIZADOS [19]-[22]

Nombre	Vel. de CPU [MHz]	Mem. Prog. [kB]	RAM [kB]	Radio freq. [GHz]	Vel. Tx [kbps]	Chip	Pot. Tx
TelosB	16	48	10	2,4	250	CC2420	0 dBm
Imote2	13-416	32 MB	256	2,4	250	CC2420	0 dBm
Zolertia	16	16 MB	96	2,4	250	CC2420	0 dBm

Se seleccionaron modelos de mota que funcionan con el chip CC2420, debido a que es el chip que soporta por defecto el simulador Castalia. Como se puede comprobar en la Tabla I, los parámetros de transmisión son idénticos, mientras los parámetros de almacenamiento y procesamiento de información son los que diferencian a los modelos de motas.

E. Escenarios de Simulación

Escenario A.- Los nodos sensores son colocados uniformemente y distribuidos de manera aleatoria, con el nodo fuente ubicado en el extremo izquierdo del escenario de simulación.

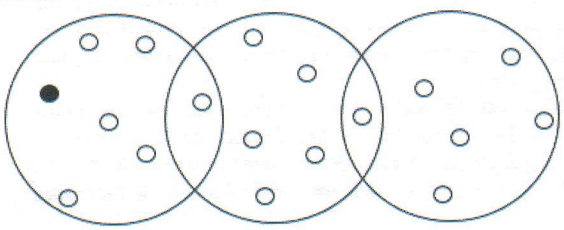


Fig. 8. Ejemplo didáctico de la disposición de los nodos sensores en el escenario "A", nodo fuente resaltado en negro [25].

Escenario B.- Los nodos sensores son colocados uniformemente y distribuidos de manera aleatoria, con el nodo fuente ubicado en el centro del escenario de simulación.

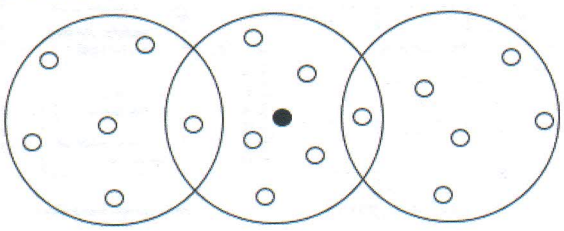


Fig. 9. Ejemplo didáctico de la disposición de los nodos sensores en el escenario "B", nodo fuente resaltado en negro [25].

IV. RESOLUCIÓN DEL TSP: MÉTODO HEURÍSTICO

En un artículo publicado por la Universidad de Buenos Aires en [24] se define el término *heurístico* en ciencias computacionales como un tipo de algoritmo que, ignorando cierta información, se libra de gran parte del esfuerzo que debió haberse requerido para leer los datos y hacer cálculos con ellos. Por otra parte, la solución producida por tal heurística, es independiente de la información ignorada, y de este modo no se ve afectada por cambios en tal información. Idealmente, ignora información que resulta muy caro colectar o mantener o que contribuye en menor grado a la precisión de la solución. Al ser el TSP un problema del tipo NP, uno de los flancos por los que se ataca la resolución del mismo consiste en un conjunto de técnicas heurísticas, de las cuales se toma la resolución del MST como una de ellas.

A. Árbol de expansión mínima

Un árbol de expansión de un grafo conexo consiste en un subgrafo que contiene todos los nodos del grafo y no tiene ciclos. El árbol de expansión mínima de un grafo pesado no dirigido, es el árbol de expansión cuyo peso (la suma de los pesos de todas sus aristas) no es mayor al de ningún otro árbol de expansión. De hecho, el problema de encontrar el MST de un grafo pesado no dirigido arbitrario tiene una gran cantidad de aplicaciones importantes y se dispone (desde 1920) de algoritmos para encontrarlo; sin embargo, la eficiencia de las implementaciones varía ampliamente y los investigadores aún buscan mejores métodos [25]

B. Comparación de los algoritmos de Prim, Borůvka y Kruskal para obtener el MST en el menor tiempo posible.

El algoritmo de Prim en su versión básica es el más fácil de implementar y por ende es el menos complejo ya que no necesita estructuras de datos complejas como Borůvka, el cuál utiliza listas o Kruskal que usa unión-find. Debido a que según [26] Zolertia, según [27] TelosB y según [28] Imote2 soportan el lenguaje de programación C, el uso de un algoritmo que requiera estructuras de datos complejas podría complicar el desarrollo de la solución. La dificultad dependerá del ingenio y la experiencia del programador, ya que sí se pueden construir estructuras de datos en C, tal como se puede apreciar en [29] y la bibliografía citada en el mismo.

La complejidad de ejecución del algoritmo y su complejidad en tiempo resultaron ser inversamente proporcionales para el caso del algoritmo de Prim, ya que si se utiliza la estructura de datos "heap" (*pila),* su complejidad en tiempo es la misma que la de los algoritmos de Borůvka y Kruskal. Es decir: $O_{(m \log n)}$ donde m es el número de iteraciones y n es el número de vértices. Con respecto al algoritmo de más rápida ejecución, [30] establece un análisis de los 3 algoritmos en cuestión (usando el algoritmo de Prim en su versión básica) y los resultados obtenidos se resumen en la Fig. 10.

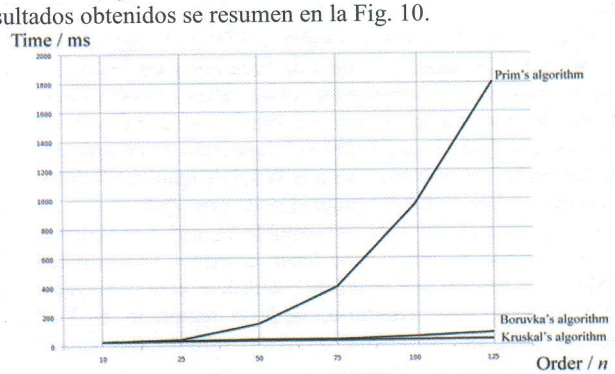


Fig. 10. Número de nodos v.s. Tiempo de ejecución para los algoritmos de Prim (en su versión básica), Borůvka y Kruskal, según [20].

Como se puede observar en la Fig. 10, cuando el número de nodos es menor a 25 los tres algoritmos tienen un comportamiento similar. Sin embargo, el algoritmo de Prim tiene ventaja en su versión básica por ser el de menor complejidad. Por lo tanto, al definir el uso de una WSN de 18 nodos sensores en el presente trabajo, se utiliza el algoritmo de Prim en su versión básica como el que mejor se adapta a los requerimientos de la implementación propuesta y con el algoritmo en cuestión se procederá a resolver el TSP de forma heurística.

C. MST y 2-opt para resolver el TSP

Para Dallaali en [23], se pueden tomar dos caminos para resolver heurísticamente el TSP:

a) Uso del MST como punto de inicio.
b) Uso del MST como una "buena" solución factible.

Luego al recorrer todas las posibles permutaciones factibles:

c) Las permutaciones deben convertir el MST en un tour del grafo original (G).
d) Las restricciones del subtour deben ser revisadas siempre, para asegurarse de que las soluciones son factibles. En otras palabras, las restricciones del subtour deben ser satisfechas.

Se toma el MST como punto de inicio, y luego se aplica el algoritmo 2-opt el cual dice que, si un ciclo Hamiltoniano se cruza a sí mismo, puede ser fácilmente acortado, basta con eliminar las dos aristas que se cruzan y reconectar los dos caminos resultantes mediante aristas que no se corten, obteniendo un ciclo final es más corto que el inicial. Un movimiento 2-opt consiste en eliminar dos aristas, para posteriormente conectar los 4 vértices que quedaron desconectados de una forma distinta, obteniendo una nueva ruta.

En Depth Last [21], se demuestra que para un grafo G, el tour que resuelve el TSP de forma aproximada en G, es menor o igual al doble del costo total del MST en G. Es por ello que tomando como punto de partida el MST se va aplicando el algoritmo 2-opt validando que se vaya construyendo un tour que resuelva el MST y se toma como solución el primer valor de costo total del tour modificado que sea menor al costo de recorrer el MST dos veces.

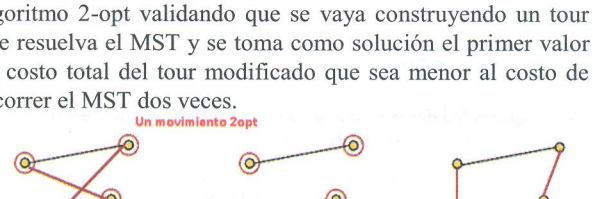


Fig. 11. Algoritmo 2opt para realizar permutaciones, en busca del tour óptimo [22].

V. RESOLUCIÓN DEL TSP: MÉTODO EXACTO'

Según [23], Branch and Bound (B&B) es un bien conocido método de búsqueda que tiene sus orígenes en el trabajo realizado sobre el TSP. B&B está organizado de manera que realiza una búsqueda exhaustiva de la mejor solución en un conjunto especificado. Cada paso de ramificación divide el espacio de búsqueda en uno o más subconjuntos, en un intento de crear subproblemas que pueden ser más fáciles de resolver que el original. Mediante la realización repetitiva de pasos de ramificación se crean una colección de subproblemas que necesitan ser resueltos, cada uno de ellos definidos por un subconjunto de tours que incluyen ciertas aristas y excluyen ciertas otras.

Antes de buscar un subproblema y posiblemente continuar dividiéndolo, un límite es computado sobre el costo de los tours. El propósito del paso de acotación es intentar evitar una búsqueda infructuosa de un subproblema que no contiene una solución mejor de aquella que ya ha sido descubierta. La idea consiste en que, si el límite es más grande que o igual a el costo de un tour que ya hemos encontrado, entonces podemos descartar el subproblema sin ningún peligro de perder el mejor tour.

A. El límite inferior de Held - Karp

Según una publicación de la Universidad George Washington en [22], el mejor algoritmo óptimo conocido, es el algoritmo de Held – Karp. Esto se comprueba con lo enunciado en [24], donde se afirma que el límite inferior de Held – Karp es usado para juzgar el desempeño de cualquier nueva solución heurística propuesta para resolver el TSP. Debido a este nivel de precisión que presenta la relajación del TSP mediante el límite inferior de Held –Karp, se ha escogido este límite inferior como la cota inferior para la resolución del TSP usando el método de Branch and Bound.

Al considerar un grafo cuyos vértices van numerados del cero al "n", de tal forma que se define un 1-tree (one tree) como un subgrafo formado por un árbol de expansión de vértices del uno al "n" y las dos aristas de menor costo que provienen del vértice cero. Es importante señalar que cada tour (incluyendo el tour óptimo) es un 1 – tree. El 1-tree mínimo se define como el 1 – tree de menor costo entre todos los 1 – tree posibles para el grafo en cuestión.

Fig. 12. Resumen de la elaboración de un 1 – tree [22].

La idea del límite inferior de Held – Karp consiste en poder alterar el grafo original (G), de tal manera que el grafo modificado (G') genere diferentes mínimos 1-tree que el grafo original. La manera en que se modifica al grafo original (G) para obtener el grafo modificado (G') se lo realiza mediante la adición de pesos a los vértices del grafo original, los pesos a añadirse son calculados reiteradas veces utilizando el método del subgradiente, de tal manera que este límite inferior se va perfeccionando.

En una primera aproximación, después de haber obtenido la matriz de costos correspondiente al grafo conformado por los nodos de la WSN a simularse, el siguiente paso es encontrar el costo total de un mínimo 1-tree sin usar Held-Karp (es decir, sin que los nodos posean pesos asociados). Cuando se inicializan las variables del algoritmo, el límite inferior óptimo

por defecto es infinito. El objetivo de encontrar el mínimo 1-tree, sin usar Held-Karp, es establecer el primer límite inferior aproximado con el que se pueda empezar a trabajar, ya que un mínimo 1-tree es por defecto un límite inferior en la resolución del TSP. Cabe recalcar que además de no trabajar con Held-Karp en este punto, tampoco se empieza la puesta en marcha del algoritmo de Branch and Bound por el momento.

Para entender la segunda aproximación, se debe tener en cuenta que al término de cada iteración de Held-Karp se obtiene un nuevo límite inferior, que inmediatamente es evaluado con dos criterios propios del algoritmo de B&B:

1) ¿El límite inferior obtenido con Held-Karp es mayor que el mejor límite inferior obtenido hasta ahora?

De ser afirmativa la respuesta, no vale la pena seguir iterando con Held-Karp, ya que este problema o subproblema de B&B no nos conduce a la solución óptima, la cual busca el mínimo costo total con el que se resuelve el TSP. Si la respuesta es negativa se continúa con la segunda pregunta a continuación.

2) ¿Se puede mejorar el límite inferior obtenido en la última iteración?

En este punto se sabe que se está trabajando sobre un límite inferior que puede o no ser inferior al mejor límite inferior obtenido hasta el momento. Aunque, no se conoce si en una iteración pasada de Held-Karp ya se obtuvo el mejor límite posible.

Si ya se obtuvo el mejor límite inferior, no se sigue variando los parámetros del método del subgradiente y se mantiene este límite hasta que el proceso de B&B determine la mejor solución al TSP, de ser negativa la respuesta, el proceso de Held-Karp continuará normalmente.

VI. SIMULACIÓN Y RESULTADOS

Todo el proceso de simulación desarrollo en Castalia y los algoritmos fueron implementados en Java, donde a continuación se presentan las gráficas obtenidas:

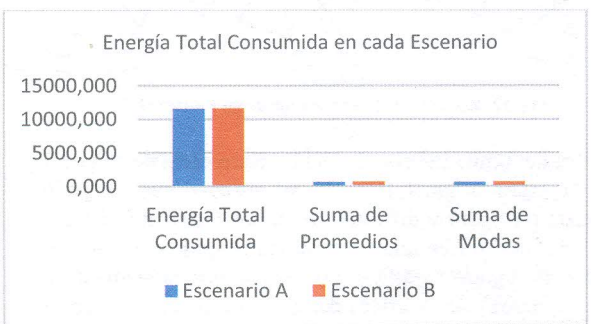


Fig. 13. Energía total consumida por la WSN, según el escenario de simulación [25].

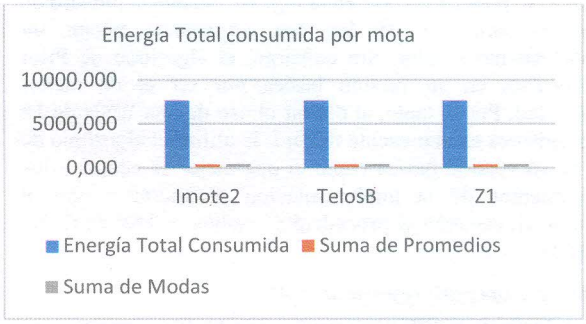


Fig. 14. Energía total consumida por la WSN, según el modelo de mota [25].

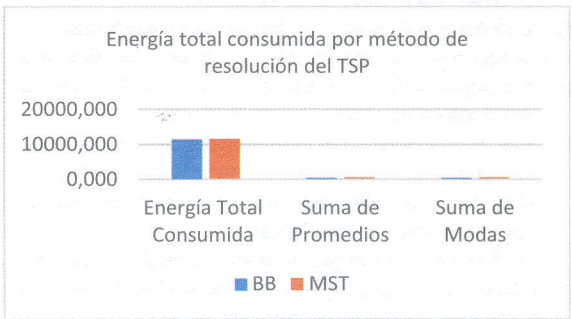


Fig. 15. Energía total consumida por la WSN, según el método de resolución del TSP [25].

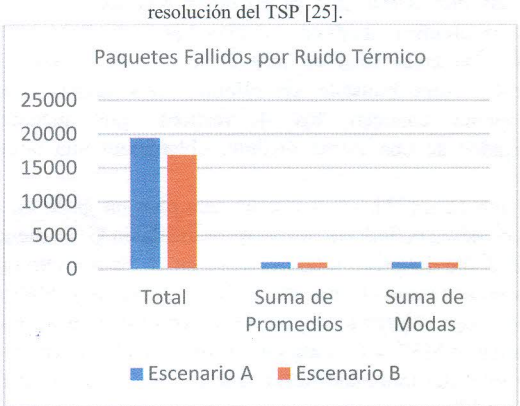


Fig. 16. Paquetes fallidos por ruido térmico en la WSN, según el escenario de simulación [25].

VII. CONCLUSIONES

El método por B&B utilizando el límite inferior de Held-Karp se tarda 1 segundo en resolver el TSP, mientras el método aproximado por el MST tarda 12 segundos en llevar a cabo la misma tarea, por lo tanto, a pesar de que B&B es más complejo de implementar, también es más rápido, aunque esto dependerá de las facilidades de programación que se tenga en la WSN específica.

Con la resolución del TSP se comprueba que la minimización del consumo energético en la WSN se cumple al observarse que cada nodo consume alrededor de 53 Joules, en llevar a cabo la implementación el TSP para todos los casos de WSN planteadas. [25].

Debido a que la resolución del TSP debe cumplir con la condición de pasar por todos los nodos antes de regresar al nodo fuente utilizando el camino de menor costo posible, para grafos no muy conexos existe una fase en la resolución del TSP para

ambos algoritmos, donde se llega a un punto en el que no se puede escoger otro camino más que el enlace infinito. Por lo tanto se concluye existen WSN en las que el TSP no podrá ser resuelto

A causa de las permutaciones introducidas por el algoritmo 2-opt, necesarias para obtener la resolución del TSP utilizando el método aproximado por el MST, el desempeño obtenido por este método es muy inferior al desempeño obtenido por el método de B&B.

VIII. Referencias

[1] Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow y P. Polakos, «Wireless Sensor Network Virtualization: Early Architecture and Research Perspectives,» IEEE Network, p. 104, 2015.

[2] G. Tirado, El problema del viajante con múltiples pilas, Madrid: Universidad Complutense de Madrid, 2011, p. 2.

[3] O. Jumira y S. Zeadally, FOCUS Series, Volume 1 : Energy Efficiency in Wireless Networks, Somerset: John Wiley & Sons, 2013, p. 29.

[4] V. Voloshin, Introduction to graph theory, Nueva York: Nova, 2009, pp. 14-15.

[5] L. Yifu, «Traveling Salesman Problem (TSP). An Example A truck needs to leave node 1, visit each of the other nodes one and only one time, and back to node 1 –What.,» Diciembre 2015. [En línea]. Available: http://slideplayer.com/slide/8783733/. [Último acceso: 19 Junio 2016].

[6] U. Colesanti y S. Santini, A Performance Evaluation Of The Collection Tree Protocol Based On Its Implementation For The Castalia Wireless Sensor Networks Simulator, Zurich: ETH Zurich, 2010, p. 2.

[7] S. Chen, M. Coolbeth, H. Dinh, . Y.-A. Kim y B. Wang, Data Collection with Multiple Sinks in Wireless Sensor Networks, Connecticut: University of Connecticut, 2009.

[8] W. Dargie y C. Poellabauer, Fundamentals of Wireless Sensor Networks (Theory and Practice), Singapur: Wiley, 2010, pp. 47-51.

[9] I. Akyildiz y M. Can Vuran, Wireless Sensor Networks, Singapur: Wiley, 2010, pp. 2-15.

[10] D. C. Herrera, Evaluación de redes de sensores inalámbricos mediante el Simulador OMNeT++, Valencia: Universidad Politécnica de Valencia, 2014, pp. 11-12.

[11] A. Boulis, Castalia: A simulator for Wireless Sensor Networks and Body Area Networks, Sydney: NICTA, 2011, p. 26.

[12] J. M. Linares Arenas, Simulación e implementación de una red de sensores inalámbrica multisalto para la medición de consumo energético en un edificio., Cataluña: Universidad Politécnica de Cataluña, 2014, pp. 9-12.

[13] M. A. Dallaali, Network Design and Optimization (Slide pack 6), Melbourne: The University of Melbourne, 2013.

[14] S. Ramos, «Heurísticas y Problemas Combinatorios,» Universidad de Buenos Aires, [En línea]. Available: http://webcache.googleusercontent.com/search?q=cache:http://materias.fi.uba.ar/7114/Docs/ApunteHeuristicas.pdf. [Último acceso: 16 05 2016].

[15] E. Coto, Algoritmos Básicos de Grafos, Caracas: Universidad Central de Venezuela, 2003, pp. 18-20.

[16] Zolertia, «Mainpage:Contiki Lesson 0,» Sourceforge, [En línea]. Available: http://zolertia.sourceforge.net/wiki/index.php/Mainpage:Contiki_Lesson_0. [Último acceso: 2016 05 16].

[17] SmartSantander, «TelosB,» SmartSantander, 30 11 2012. [En línea]. Available: http://smartsantander.eu/wiki/index.php/Main/TelosB. [Último acceso: 2016 05 16].

[18] K. Lorincz, «IMote2 Installation Instructions,» Intel Research, Berkeley, 14 06 2005. [En línea]. Available: http://www.eecs.harvard.edu/~konrad/projects/imote2Camera/IMote2-Installation-Instructions.html. [Último acceso: 16 05 2016].

[19] J. Straub, C Programming: Data structures and Algorithms, Washington D.C.: University of Washington, 2006.

[20] I. Podsechin, «EURAXESS,» 2008. [En línea]. Available: http://www.euraxess.fi/Tiedostot/Tiedostot/Viksu/Viksu%202008/IgorPodsechin.pdf. [Último acceso: 16 05 2016].

[21] A. Paterson, «Performing 2-OPT Updates To TSP Solution Approximation In Java,» Depth Last, 6 Abril 2016. [En línea]. Available: http://depthlast.com/2016/04/06/performing-2-opt-updates-to-tsp-solution-approximation-in-java/. [Último acceso: 12 Junio 2016].

[22] R. Simha, «The Traveling Salesman Problem (TSP),» George Washington University, [En línea]. Available: https://www.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html. [Último acceso: 11 05 2016].

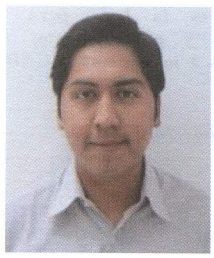
[23] L. D. Applegate, E. R. Bixby, V. Chvátal y J. W. Cook, The Traveling Salesman Problem, A Computational Study, New Jersey: Princeton University Press, 2006, pp. 41-42.

[24] D. Davendra, Traveling Salesman Problem, Theory and Applications, Rijeka: InTech, 2010, pp. 1-17.

[25] Tito Ontaneda, J. E., Comparación de los métodos MST y B&B en la resolución del TSP en una WSN simulada, 2016, EPN.

IX. BIOGRAFÍA

Jonathan E. Tito O. Estudiante del Master Universitario en Inteligencia Artificial en la Universidad Politécnica de Madrid, Ingeniero en Electrónica y Redes de Información por Escuela Politécnica Nacional en Quito. Su trabajo está basado en algoritmos de optimización, teoría de grafos y la informática aplicada a la automatización y mejora de servicios a la comunidad.



Marco E. Yacelga P. Recibió su grado de Maestría en Telecomunicaciones de la Universidad de Melbourne en Melbourne, Australia en 2014. Además, se graduó de Ingeniero en Electrónica y Telecomunicaciones de la Escuela Politécnica Nacional en Quito, Ecuador, en 2011. Sus intereses de investigación están vinculados al diseño y optimización de Redes Ópticas, Sensores Wireless y Celulares, Internet de las cosas, algoritmos de optimización de redes 5G y Redes Heterogéneas.