WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2013; **13**:1369–1385
Published online 15 August 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1184

RESEARCH ARTICLE

Energy-conserving data gathering by mobile mules in a spatially separated wireless sensor network

Fang-Jing Wu* and Yu-Chee Tseng

Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan

ABSTRACT

This paper considers a *spatially separated wireless sensor network*, which consists of a number of isolated subnetworks that could be far away from each other in distance. We address the issue of using mobile mules to collect data from these sensor nodes. In such an environment, both data-collection latency and network lifetime are critical issues. We model this problem as a bi-objective problem, called *energy-constrained mule traveling salesman problem* (EM-TSP), which aims at minimizing the traversal paths of mobile mules such that at least one node in each subnetwork is visited by a mule and the maximum energy consumption among all sensor nodes does not exceed a pre-defined threshold. Interestingly, the traversal problem turns out to be a generalization of the classical *traveling salesman problem* (TSP), an NP-complete problem. With some geometrical properties of the network, we propose some efficient heuristics for EM-TSP. We then extend our heuristics to multiple mobile mules. Extensive simulation results have been conducted, which show that our proposed solutions usually give much better solutions than most TSP-like approximations. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS
data gathering; data mule; energy conservation; mobile sensor; traveling salesman problem (TSP); wireless sensor network

***Correspondence**
Fang-Jing Wu, Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan.
E-mail: fangjing@cs.nctu.edu.tw

1. INTRODUCTION

The progress of embedded micro-sensing microelectromechanical systems (MEMS) and wireless communications has made *wireless sensor networks* (WSNs) feasible. A conventional WSN normally consists of a sink and many inexpensive sensor nodes deployed in a sensing field. Each node has the capability of sensing, collecting, processing, and storing environment information, and communicating with neighboring sensor nodes. Many applications such as object tracking, health monitoring, security surveillance, and intelligent transportation [1–4] have been proposed.

This paper considers a *spatially separated WSN* (SS-WSN), which consists of several isolated subnetworks. These subnetworks are not connected because of reasons such as cost constraints, physical constraints (rivers and mountains), or unavoidable disasters (explosions or earthquakes), and thus called spatially separated. For example, as a result of geographical constraints, the sensing field could be huge, and deploying a connected WSN is very difficult. Even if the WSN is initially connected, it could be partitioned because of emergencies such as fires. In addition, when random deployment is adopted, the network is not necessarily connected. Therefore, we believe that an SS-WSN may appear frequently in many practical applications, such as ecology-observing systems [5]. However, coordination among these isolated subnetworks is necessary. We thus consider using mobile mules to travel among these subnetworks to collect data. Figure 1 shows an example, where a helicopter serves as a mobile mule to provide connectivity among these isolated subnetworks. In fact, depending on different scenarios and availability of mules, there may exist various applications of such an SS-WSN architecture. For example, in a long-term monitoring application of a huge forest, deploying an SS-WSN is inevitable. In this case, a set of robotic cars may patrol along footpaths in the forest to collect sensing data of multiple geographical areas, where the moving paths between two subnetworks can be simply modeled as a set of footpaths. In contrast, in underwater monitoring applications, although there is no geographical constraints in an underwater environment, data collection may highly rely on mobile mules because dramatic signal attenuation can easily partition

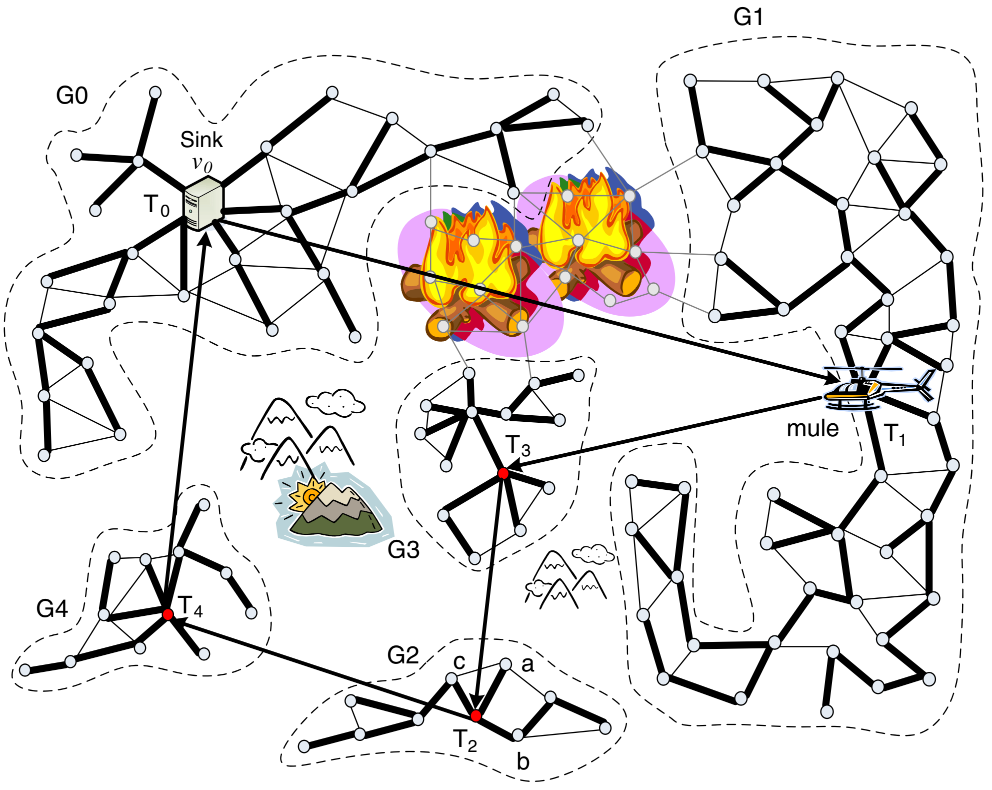


Figure 1. An example of one round of data gathering by a mobile mule in a spatially separated wireless sensor network.

a sensor network. In this case, submarines may serve as mobile mules [6].

This paper considers the data gathering issue in an SS-WSN. We focus on exploiting mobile mules ('mules' for short) to collect subnetworks' sensory data. For example, in Figure 1, a mule is initially located at the sink node v_0. Assuming an identical data generation rate for each sensor, our goal is to dispatch the mule from v_0 to visit one sensor node in each subnetwork and then return to v_0. The node being visited in each subnetwork is called the *landing port* of the subnetwork. Any node can be selected as a landing port. (For load balance consideration, it is possible to have multiple landing ports in a subnetwork; this will be addressed later on.) Therefore, the mule will switch between an inter-subnetwork *movement state* and an intra-subnetwork *data gathering state*. During the movement state, the mule leaves from its current landing port and moves to the next landing port. During the data gathering state, the mule stays at a landing port, contacts all one-hop neighboring nodes (termed *gateway nodes*), requests all nodes in this subnetwork to form a data-collection tree rooted at the mule, and commands all nodes to relay their sensory data to the mule along the tree. When the mule returns to v_0, it relays all collected data to v_0. This completes one round of data gathering.

Two critical issues in the aforementioned scenario are *data-collection latency* and *energy conservation*. The former is to quickly collect sensory data from all sensors in the SS-WSN, whereas the latter is to prolong the network lifetime. To minimize the data-collection latency, we need to compute the shortest traversal path for the mule to visit each subnetwork in exactly one landing port. To prolong the network lifetime, we should enforce the mule to visit each node in each subnetwork (implying that a node needs not to relay data for others). Clearly, these are trade-offs. In this paper, we define a new problem called *energy-constrained mule traveling salesman problem* (EM-TSP), which is a generalization of the Euclidean traveling salesman problem (ETSP) [7]. The goal of EM-TSP is to find the shortest traversal path of the mule to visit each subnetwork *at least* one landing port. The landing ports of each subnetwork will connect all nodes in that subnetwork via some data-collection trees rooted at them, such that the maximum energy consumption among sensor nodes does not exceed a pre-defined threshold τ_e. EM-TSP generalizes ETSP as follows: (i) it is sufficient to visit some landing ports in each subnetwork; (ii) load balance among gateways (due to the selection of landing ports) is essential; and (iii) the construction of intra-subnetwork data-collection trees also matters. On the other hand, if we limit τ_e such that a node can only transmit the data generated by itself but not relay data for others, EM-TSP degenerates to ETSP. We will give a formal nondeterministic polynomial (NP)-hardness proof.

Clearly, the existing solutions to ETSP cannot directly be applied to EM-TSP efficiently. We thus need to design a new data gathering scheme for EM-TSP. It consists of three phases. The first phase is to properly partition each subnetwork, if necessary, into multiple subnetworks to meet the constraint τ_e. Note that τ_e is an application-specific parameter to provide flexibility in dealing with different application requirements. For example, one may use τ_e to balance between minimizing data-collection latency and maximizing the network lifetime. When τ_e is very large, each original subnetwork will remain as one subnetwork, thus imposing more work on data relay and damaging network lifetime. On the contrary, when τ_e is very small, each original subnetwork will be divided into many small subnetworks, thus imposing more work on mules. The second phase is to plan a traversal path for the mule to visit each subnetwork. Several schemes are proposed. In particular, we adopt the approach in [8], which shows that, when nodes are placed in a Euclidean space, the CH of input nodes in ETSP has some geometrical properties closely related to the optimal solution to ETSP. This motivates us to define a special convex polygon, termed *convex container* (CC), with some geometrical properties that help find thes shorter traversal path. Finally, the third phase is to construct a data-collection tree in each subnetwork to minimize the energy consumption of nodes. In addition, we also extend our proposed scheme to the case of multiple mules. Extensive simulations have been conducted, which show that an exhausted search usually cannot find the optimal solution in a limited time, whereas our heuristic not only has low computation complexity but also gives much better solutions than several TSP-like approximations.

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 describes our network model and the problem definition. Section 4 presents our algorithms. Simulation results are in Section 5. Finally, Section 6 concludes this paper.

2. RELATED WORK

In this section, we survey some mobility control schemes in mobile WSNs [9] and some path-planning techniques.

For mobility control, researchers have studied the functionality of mobile sensor nodes in both *homogeneous* and *heterogeneous* networks. For homogeneous networks, the authors of [10–12] consider that all sensor nodes have identical capability. To address coverage and connectivity issues, Zou and Chakrabarty [10] and Heo and Varshney [11] adopt *virtual forces* to move sensors, whereas Wang *et al.* [12] use *Voronoi diagram* to detect coverage holes and then move sensors to cover these holes.

For heterogeneous networks, it is normally assumed that there are some resource-richer mobile nodes. In [13] and [14], *actors* are used to respond to dynamic events and provide appropriate actions. In [15] and [16], mobile actors with long transmission ranges are used to relay data for their local cluster, which are formed by static sensors. Recently, for spatially separated subnetworks, researchers have designed mobile nodes, called *data mules*, to conduct message relaying. Random mobility are assumed for such mules in [17–19]. Because random mobility may incur unbounded data-collection delay, *message ferries* with controllable mobility are studied. Reference [20] designs moving paths of robots for search and rescue systems. In [21], there is a ferry moving along a publicly known route. With knowledge of the ferry route, nodes can proactively schedule their transmission/reception with the ferry. An optimization problem is to find a ferry route such that the average message delay is minimized and that the communication time of each node can be met. Two ferrying schemes are proposed in [22]. One allows nodes to periodically move closer to the ferry route. The other allows nodes to request the ferry, via high-power radios, to approach them when they have intention to transmit/receive. The optimization goal is to minimize the message drops. Multiple ferries are considered in [23], where packets may be relayed by multiple ferries before reaching their destinations. In [24], sensors with different weights are deployed sparsely in an isolated way. Data mules must visit sensors along deterministic paths to collect their data such that sensors with higher weights are visited with lower inter-arrival time and that the total length of paths is minimized. Reference [25] further extends [24] by designing probabilistic paths for data mules. In [26], sensors are modeled as disjoint disks with different radii, and a mobile robot is dispatched to visit each sensor with the shortest traversal path. Note that a sensor is visited once the robot is within its communication range. (In comparison, our work assumes that it is sufficient to visit one representative node in each subnetwork.) To relieve the *funneling effect* [27] in a connected WSN, researchers have proposed to use mobile collectors. Given a candidate set of *rendezvous points*, finding moving paths to visit these points are studied in [28]. How to find better rendezvous points is studied in [29,30]. Distributed protocols are designed in [31,32] to navigate these mobile collectors, where a point may be visited by a multi-hop path.

The aforementioned results are highly related to the classical path-planning issues, such as the traveling salesman problem (TSP). In [33], a 2-approximation algorithm with complexity $O(n^2)$ for TSP based on the minimum spanning tree (MST) of nodes is proposed, where n is the number of nodes. In [34], a (1.5)-approximation, termed *Christofides heuristic*, with complexity $O(n^3)$ based on MST and the minimum-length matching of nodes is proposed. These two algorithms may find a self-intersecting path. When nodes are scattered in an Euclidean space, for a fixed $c > 1$, Arora [35] proposed a $(1 + 1/c)$-approximation scheme with complexity $O(n(\log n)^{O(c)})$, by putting nodes in a square box, partitioning them into smaller grids, and finding an initial path that enters and exits grids only through some special points, termed *portals*, such that self-intersections are avoided. Note that these approaches cannot handle EM-TSP because a mobile sensor only needs to be within the communication range of a node or a representative node of a subnetwork. With these path-planning technologies, Meliou *et al.* [36] designed the paths for query/reply messages in a static and connected WSN, where a path is allowed to split into multiple ones and eventually converge into one before arriving at the sink. Reference [37] extends [36] by considering the set of queried sensor nodes being time varying.

3. NETWORK MODEL AND PROBLEM DEFINITION

The SS-WSN is modeled as an undirected graph $G = (V, E)$, where V is the set of sensor nodes and E is the set of communication links. A special node $v_0 \in V$ is designated as the sink node and is responsible for collecting sensory data from all other nodes in V. Each node has a communication range of r_c. Constrained by r_c and the physical environment, the network G is spatially separated in the sense that it is partitioned into multiple connected *subnetworks* $G_0, G_1, \ldots, G_n$. Without loss of generality, we let sink $v_0 \in G_0$. The location of each $v_i \in V$ is denoted by (x_i, y_i) (how to identify nodes' locations is out of the scope of this work). The Euclidean distance between two v_i and v_j in V is denoted by $d(v_i, v_j)$. We assume a long-term monitoring application, where each sensor node has an identical data generation rate λ and an identical initial energy E_{total}. A *mobile mule m* is responsible for moving to these subnetworks to collect and deliver their sensing data to v_0. We assume that m also has a communication range of r_c.

We are interested in the *data gathering* issue in an SS-WSN G. This is achieved by the cooperation among v_0, m, and all subnetworks. The problem is formulated as follows. Time is divided into *rounds*. In each round, m will leave from v_0, visit each subnetwork, collect all data therein, return to v_0, and forward the collected data to v_0. Therefore, during a round, m will switch between an inter-subnetwork *movement state* and an intra-subnetwork *data gathering state*. A movement state starts when m leaves from the landing port of its current subnetwork and ends when m arrives at the landing port of the next subnetwork. A *landing port* of a subnetwork can be *any* sensor node in the subnetwork. In the beginning of a round, m will stay at v_0. Then, it will visit one or multiple landing ports in each $G_i \neq G_0$ and return to v_0. (The reason for requiring visiting multiple landing ports in G_i will become clear later on). This completes one round of data gathering. A data gathering state starts after m arrives at a landing port in G_i and ends once it has collected all nodes' newly generated sensory data. Once landed, m should contact all neighboring nodes within its communication range (termed *gateway nodes*), request all nodes in G_i to form a *data-collection tree* rooted at m, and instruct all nodes in the tree to relay their sensory data to m along the tree. Note that the mule does not need to conduct data collection in G_0 because nodes in G_0 can report to v_0 at any time. Figure 1 gives an example, where sensory data of each G_i is relayed though its tree to the mule and then to v_0, except G_0.

We make two notes as follows. First, for practical reasons, when a subnetwork contains too many nodes, we may enforce it to be divided into multiple subnetworks, thus requiring multiple landing ports. This may reduce the energy consumption of gateway nodes and balance their load. We will show how to conduct such partitioning later on. As follows, for ease of presentation, unless stated otherwise, a 'subnetwork' will refer to one after conducting such partitioning. Second, after m landed at a landing port in G_i, the relaying load of this landing port should be regarded as zero (or near zero) because the root of the data-collection tree is the mule, rather than this landing port (sending its data to m is negligible). Thus, we will calculate the energy consumption of those gateway nodes associated with the tree. For example, in Figure 1, gateway nodes a, b, and c will take care of sensory data of 1 node, 3 nodes, and 5 nodes, respectively.

We consider two main performance metrics: *data-collection latency* of m and *energy consumptions* of sensor nodes. The former is modeled by the length of the traversal path of m in a round, whereas the latter is modeled by the maximum energy consumption among all sensor nodes in a round. (We do not consider the data-collection latency per subnetwork because it should be relatively much faster than the movement of the mule.[†]) By putting these two goals together, the objective in a round becomes finding the shortest traversal path of m to visit each subnetwork such that the maximum energy consumption among sensor nodes is minimized. However, these two goals contradict each other. Visiting a subnetwork exactly once is the best for the first metric but the worst for the second metric and vice versa. To resolve this problem, we define an optimization problem, termed EM-TSP, where the goal is to find the shortest path for m to visit each subnetwork at least once and then return to v_0 such that the maximum energy consumption among all sensor nodes does not exceed than a threshold τ_e. To measure the energy consumption of a node, we consider the data-collection tree when m visits a subnetwork G_i. Let T be the data-collection tree of G_i. We model the energy cost of $v_k \in G_i$ by $E_T(v_k) = e \cdot \lambda \cdot \delta \cdot |T(v_k)|$, where e is the energy consumption for a node to transmit one unit of sensory data, λ is the data generation rate of each sensor node, δ is the maximum duration of a round, and $T(v_k)$ is the subtree of T rooted at v_k. That is, $E_T(v_k)$ includes the energy cost to report the sensory data of v_k and v_k's descendants.

[†]Let Δt be the packet transmission time on a link. Given a subnetwork G_i, if a well-scheduled MAC protocol [38] is adopted, the intra-subnetwork data-collection latency can be approximated by $\Delta t|G_i|$ (this happens when a pipeline effect occurs such that one packet is delivered to the mule per time unit while the mule is visiting G_i). Therefore, the total data-collection latency can be approximated by the mule traveling time plus $\sum_{i=0}^{n} \Delta t|G_i|$. The latter factor is close to a constant given a fixed G.

Definition 1. *Given an SS-WSN $G = (V, E)$, a sink node v_0 in G_0, and an energy threshold τ_e, the EM-TSP is to find a proper partition of G into subnetworks and a traversal path P starting and ending at v_0, visiting each subnetwork $G_i \neq G_0$ in one landing port, and connecting all nodes in G_i via a data-collection tree T rooted at the landing port, such that $\max_{v_k \in V} E_T(v_k) \leq \tau_e$ and the total length $|P|$ is minimized.*

To prove that EM-TSP is NP-hard, we define a decision problem as follows.

Definition 2. *Given an SS-WSN $G = (V, E)$, a sink node v_0 in G_0, an energy threshold τ_e, and a positive integer L, the length-constrained and energy-constrained mule TSP (LEM-TSP) is to find a proper partition of G into subnetworks and a traversal path P starting and ending at v_0, visiting each subnetwork $G_i \neq G_0$ in one landing port, and connecting all nodes in G_i via a data-collection tree T rooted at the landing port, such that $\max_{v_k \in V} E_T(v_k) \leq \tau_e$ and $|P| \leq L$.*

Theorem 1. *Length-constrained and energy-constrained mule traveling salesman problem is NP-hard.*

Proof of Theorem 1 is proved in Appendix 6. We reduce ETSP [7], an NP-hard problem, to a special case of LEM-TSP by regarding each subnetwork as a 'macro' node.

4. HEURISTICS TO ENERGY-CONSTRAINED MULE TRAVELING SALESMAN PROBLEM

In this section, we propose some heuristics to solve EM-TSP. Our solutions consist of three phases: (i) *subnetwork partition*; (ii) *path planning*; and (iii) *balanced tree construction*. The first phase partitions each 'original' subnetwork,[‡] if necessary, into multiple smaller subnetworks such that each subnetwork can meet the energy requirement τ_e. The second phase is to plan a traversal path of m to visit each subnetwork at one landing port such that the total path length is as small as possible. We will propose three schemes for phase 2. The third phase is to form a data-collection tree rooted at each landing port such that the minimum remaining energy among sensors is maximized. At the end, we will analyze the complexity of our heuristics and discuss how to extend to multiple mules.

4.1. Subnetwork partition

To meet the energy constraint τ_e, this phase tries to partition each original subnetwork into several subnetworks such that (i) the amount of sensory data relayed by each sensor is bounded and (ii) the number of subnetworks after partitioning is minimized. To achieve these goals, we first try to limit the number of sensors in each subnetwork after partitioning within a bound $\tau_s = \lfloor \frac{\tau_e}{e \cdot \lambda \cdot \delta} \rfloor$ such that each original subnetwork G_i is partitioned into $\alpha_i = \lceil \frac{|G_i|}{\tau_s} \rceil$ subnetworks. In the following, we propose a modified k-means algorithm to solve this problem. Note that the typical k-means algorithm [39] cannot properly handle this partitioning problem for two reasons. First, it cannot guarantee the number of nodes in each set. Second, we require that each subnetwork is connected by itself (i.e., without passing other subnetworks). For example, in Figure 2, although the partitioning is perfect, the right subnetwork needs to rely on the left subnetwork to become connected.

Our scheme works as follows. For each original subnetwork G_i, we set α_i as its ideal number of partitions and initially partition G_i into α_i groups by the following *grouping process*. Then, we check whether each group $\mathcal{G}_j$ has $|\mathcal{G}_j| \leq \tau_s$. If not, we increase α_i by one and repeat the grouping process until each group $\mathcal{G}_j$ satisfies $|\mathcal{G}_j| \leq \tau_s$. Otherwise, each group $\mathcal{G}_j$ is regarded as a new subnetwork, and this phase terminates. Note that, after this phase, we will use $\mathcal{G}_j$ to denote a subnetwork later. The details of the grouping process are as follows.

(1) For each G_i, we randomly select α_i nodes as its initial seeds. Each seed is considered as a trivial tree.
(2) From the tree rooted at each seed, we try to grow the tree by one hop based on the breadth-first search order. For each node that has not joined any tree yet and is one hop away from at least one tree, it joins the smallest tree among all candidates.
(3) Step 2 is repeated until each node has joined a tree.
(4) For each tree, generate a new seed by choosing the node nearest to the tree's center-of-gravity.
(5) With these new seeds, re-run steps 2–4, until there is no or very little change on the sizes of these trees (a threshold may be set so that this step terminates).

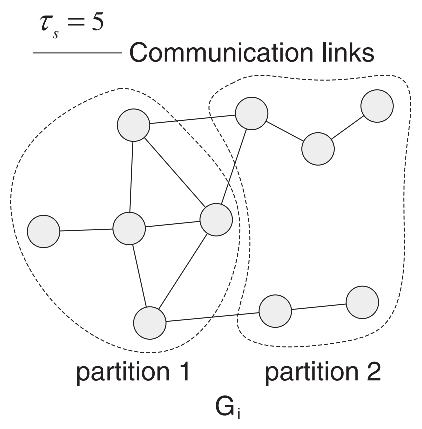


Figure 2. An example of subnetwork partition using a typical k-means algorithm.

[‡]We use 'original' subnetworks to distinguish from those after partitioning.

(6) Check if the α_i trees meet our balancing criteria (i.e., τ_s). If so, the algorithm terminates; otherwise, increase α_i by one and repeat steps 1–5 again.

4.2. Path planning

In this phase, we propose three heuristics to find a traversal path for m. The input is a set of subnetworks $\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \ldots, \mathcal{G}_q\}$. The goal is to minimize the path length. The first heuristic greedily chooses the next landing port repeatedly. The second one is derived from the CH approach in [8]. The third one relies on a special convex polygon that may lead to an even shorter path.

4.2.1. Greedy scheme.

This scheme is mainly designed for making comparisons. It works in an iterative manner. Initially, m is located at v_0. In each iteration, m chooses the next landing port, denoted by v_l, to be visited such that v_l is located at an unvisited subnetwork and closest to the current location of m. This process is repeated until all the subnetworks are visited. Finally, m returns to v_0. This finds the traversal path P.

Figure 3 gives an example of this scheme. It is to be noted that the traversal path may have intersections, which should be avoided, as to be shown later.

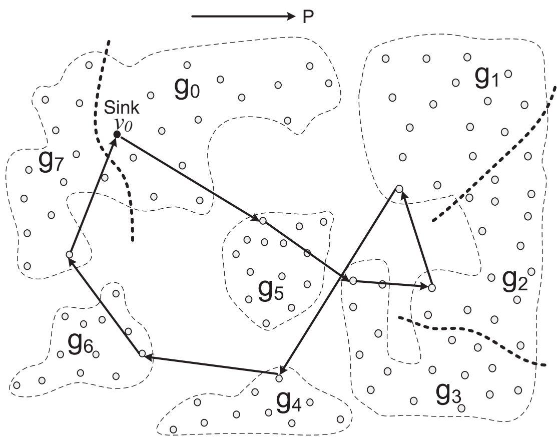


Figure 3. An example of the greedy scheme.

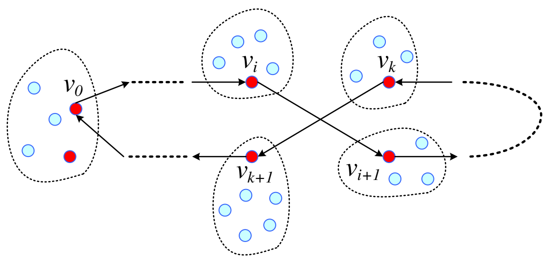


Figure 4. The proof of Theorem 2.

4.2.2. Convex hull-based (CH-based) scheme.

This scheme first selects a delegation node in each subnetwork and then constructs a *convex hull* (CH) of these delegation nodes to provide a base for constructing a traversal path such that intersections can be avoided. To start with, we extend a property raised in [40] to the following property.

Theorem 2. *An optimal traversal path to EM-TSP has no intersection with itself.*

Proof. Referring to Figure 4, an intersection is defined as two line segments that cross each other in a 2D plane. We assume that path $P = v_0 \to \cdots \to v_i \to v_{i+1} \cdots \to v_k \to v_{k+1} \to \cdots \to v_0$ is an optimal solution to EM-TSP such that $\overline{v_i v_{i+1}}$ and $\overline{v_k v_{k+1}}$ intersect each other. However, we can find another traversal path $P' = v_0 \to \cdots \to v_i \to v_k \to \cdots \to v_{i+1} \to v_{k+1} \to \ldots \to v_0$ such that $|P'| < |P|$ by the triangle inequality. It is a contradiction. Therefore, this theorem is proven. □

A CH has some good geometric properties that can help find a shorter traversal path. First, a CH is a convex polygon that never intersects with itself. Second, Larson and Odoni [40] proved that the order of boundary nodes on the CH must appear, in that order, in the optimal solution to ETSP. These observations motivate us to conduct path construction and path improvement based on a CH. In the following, we modify the algorithm in [8] into one fitting our need.

(1) For each subnetwork $\mathcal{G}_i \neq \mathcal{G}_0$, compute a delegation node d_i, where d_i is the node closest to the center-of-gravity of $\mathcal{G}_i$, that is, $\left(\frac{\sum_{(x_k,y_k)\in\mathcal{G}_i} x_k}{|\mathcal{G}_i|}, \frac{\sum_{(x_k,y_k)\in\mathcal{G}_i} y_k}{|\mathcal{G}_i|}\right)$. The landing port of $\mathcal{G}_i$ will be d_i.
(2) Let $D = \{v_0\} \cup \{\forall d_i | \mathcal{G}_i \neq \mathcal{G}_0\}$. Following Theorem 2, we construct a CH of D (such algorithms can be found in [33]). Let the CH be our initial path P.
(3) We iteratively add nodes in $D - P$ into the traversal path. Specifically, in each iteration, for each node $d_i \in D - P$, we try to insert d_i into each link (x, y) of P. The insertion cost of putting d_i between x and y is $cost(x, d_i, y) = d(x, d_i) + d(d_i, y) - d(x, y)$. Let $d_{\min}$ be the node whose insertion between link (x, y) incurs the smallest $cost(x, d_{\min}, y)$. Then, we update P by inserting $d_{\min}$ between (x, y) of P. We repeat this process until all delegation nodes are included in P. This finds the final path P.

Figure 5 gives an example of the CH-based scheme. After step 1, $D = \{v_0, d_1, \ldots, d_7\}$. After step 2, the CH of D is composed of v_0, d_1, d_2, d_4, d_6, and d_7. Step 3 first

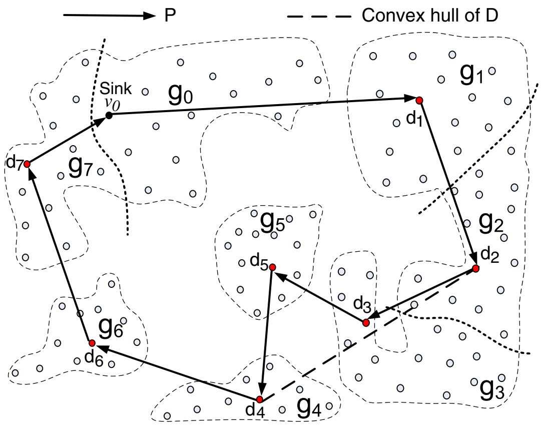


Figure 5. An example of the convex hull-based scheme.

inserts d_3 into link (d_2, d_4) and then inserts d_5 into link (d_3, d_4) to form the final P.

4.2.3. Convex container-based (CC-based) scheme.

We make two observations on the geometric properties to form a small convex polygon. First, the landing port of each subnetwork are usually closer to the inner side of the whole field. Figure 6 shows an example of an SS-WSN after conducting the subnetwork partition phase. Using outer nodes A, B, C, and D as landing ports will incur a longer path than using inner nodes A', B', C', and D'. Second, although inner nodes are generally preferred, sometimes skipping some inner nodes may even reduce the path length. In the above example, if we replace D' by D'', we can find an even better path.

The aforementioned observations motivate us to define a special kind of convex polygon, termed *convex container* (CC), that allows a subnetwork to either contribute a very inner node to the convex polygon or simply have some node(s) inside the convex polygon. The formal definition is as follows.

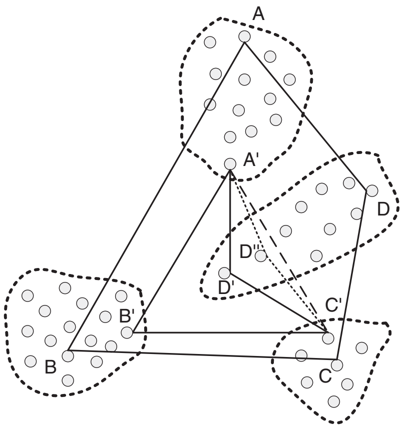


Figure 6. Examples of selecting inner nodes as landing ports.

Definition 3. *Given a set of node-disjoint subnetworks $\mathcal{G} = \{\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_q\}$, $q \geq 3$, a CC of $\mathcal{G}$ is a convex polygon $P = x_1 \to x_2 \to \cdots \to x_r$ composed of r nodes, $r \leq q$, such that for each $\mathcal{G}_i$, $i = 1, \dots, q$, either of the following conditions is satisfied:*

- *$\mathcal{G}_i$ has one node belonging to P, and this node is the only node of $\mathcal{G}_i$ contained inside P.*
- *$\mathcal{G}_i$ has no node belonging to P but has at least one node contained inside P.*

Figure 7 shows some examples of Definition 3, where there are seven subnetworks $\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_6$. The one in Figure 7(a) is not a CC because $\mathcal{G}_6$ is outside the polygon. The one in Figure 7(b) is not a CC because $\mathcal{G}_1$ has some extra nodes inside the polygon. The one in Figure 7(c) is a CC. Clearly, CCs are not unique. (Note that to take into account the case of multiple nodes forming a straight line, a node *on* P is not considered *inside* P.)

Our CC-based scheme consists of four steps. Step 1 tries to choose a 'min–max' initial node to start our construction. Step 2 is the main step to form a CC. Steps 3 and 4 add nodes of those unvisited subnetworks into the container to form a traversal path. Note that because v_0 must be included in the final path P, we will imagine that $\mathcal{G}_0$ contains only one node v_0.

(1) Let $h_0 = v_0$, and let h_i be the node in $\mathcal{G}_i$, $i = 1, \dots, q$, which has the largest y-coordinate. We then let $h_{\min}$ be the node in $\{h_0, h_1, \dots, h_q\}$ that has the smallest y-coordinate. (In Figure 8(a), $h_{\min} = h_4$.)
(2) Initially, let P contain only $h_{\min}$, and all subnetworks are considered unvisited except the one containing $h_{\min}$. We then enter an iterative *sweeping process* to add nodes into P to form a CC. Intuitively, this step simulates tying the shortest string around all subnetworks in the counterclockwise direction such that P is a CC. We imagine that there is an arrow string S of an infinite length pointing at degree 0 with $h_{\min}$ as the origin. The following steps repeatedly rotate S in the counterclockwise direction until a CC is constructed.
   (a) Rotate S, from its current direction, counterclockwise with the last node in P as its center. Stop rotating when any of the following conditions is encountered: (i) $h_{\min}$ is on string S and (ii) the first unvisited subnetwork (say $\mathcal{G}_i$) appears such that all node of $\mathcal{G}_i$ have been swept by string S and the last node/nodes swept by S is/are now on S. Note that condition (i) may happen when multiple nodes form a line. (Figure 8(b) shows an example of the ith iteration during the sweeping process.)
   (b) If $h_{\min}$ is on S, a CC P is found, and we exit this loop. Otherwise, let v_j be the node of $\mathcal{G}_i$ that is on S (if there are multiple such nodes, the one closest to the center of S is selected).

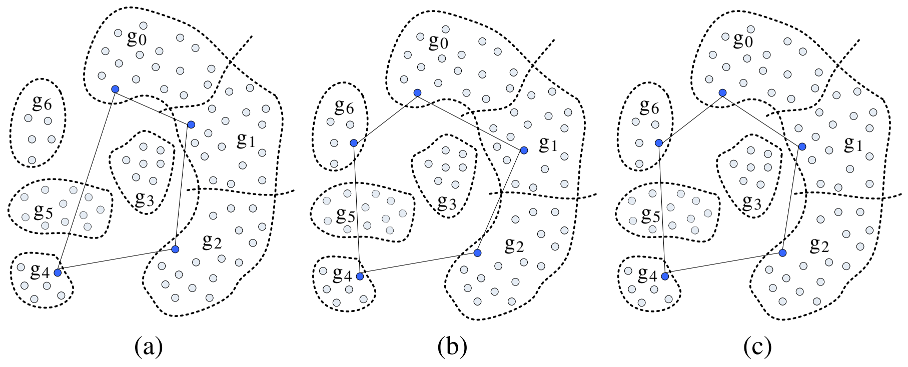


Figure 7. The polygons in (a) and (b) are not convex containers, whereas that in (c) is.

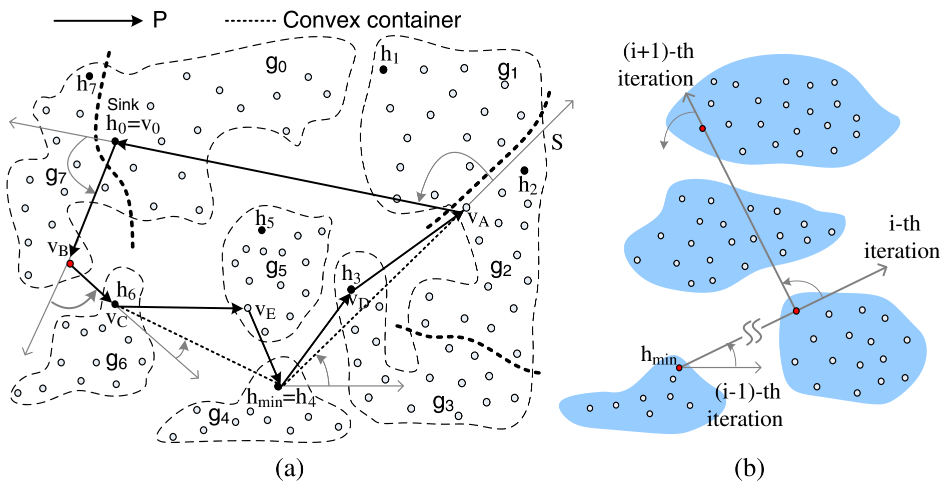


Figure 8. The CC-based scheme: (a) an example of the CC-based scheme, and (b) the i-th iteration of the sweeping process.

We then append v_j to path P, mark $\mathcal{G}_i$ as visited, and go back to step 2. (In Figure 8(a), in the first iteration, $\mathcal{G}_2$ is the first subnetwork whose nodes are all swept by S. Because the last node being swept is v_A, v_A is appended to P.)

(3) With the CC P, we divide the remaining unvisited subnetworks into two sets: $\widehat{\mathcal{G}}$ contains those subnetworks that are 'crossed' by P and $\widetilde{\mathcal{G}}$ contains those that are completely inside P. (For example, in Figure 8(a), $\mathcal{G}_3$ is 'crossed' by P, whereas $\mathcal{G}_5$ is inside P.) We first deal with set $\widehat{\mathcal{G}}$. We will iteratively choose one node in a subnetwork in $\widehat{\mathcal{G}}$ and insert it into P. Specifically, in each iteration, for each node v_j in each $\mathcal{G}_i \in \widehat{\mathcal{G}}$, we try to insert v_j between each link (x, y) of P. The insertion cost of inserting v_j into (x, y) is $cost(x, v_j, y) = d(x, v_j) + d(v_j, y) - d(x, y)$. Let $v_{\min}$ be the node among all candidates in all $\mathcal{G}_i \in \widehat{\mathcal{G}}$ that incurs the least cost. Then, we insert $v_{\min}$ into P and remove the subnetwork containing $v_{\min}$ from $\widehat{\mathcal{G}}$. This process is repeated until $\widehat{\mathcal{G}} = \emptyset$. (For example, in Figure 8(a), v_D of $\mathcal{G}_3$ is inserted into the link $(h_{\min}, v_A)$.)

(4) Next, we deal with set $\widetilde{\mathcal{G}}$. We repeat the same process as step 3 to insert more nodes into P until $\widetilde{\mathcal{G}} = \emptyset$. The final traversal path is P. (For example, in Figure 8(a), v_E of $\mathcal{G}_5$ is inserted into $(v_C, h_{\min})$.)

4.2.4. Some local optimizations.

Finally, we present some local optimization techniques to further reduce the length of P. Let $P = v_0 \rightarrow v_{p_1} \rightarrow \cdots \rightarrow v_0$ be a path obtained by any one of the above heuristics. We can examine any two consecutive links $v_{p_i} \rightarrow v_{p_{i+1}} \rightarrow v_{p_{i+2}}$, try to find another $v'_{p_{i+1}}$ which is in the same subnetwork as $v_{p_{i+1}}$ such that $d(v_{p_i}, v'_{p_{i+1}}) + d(v'_{p_{i+1}}, v_{p_{i+2}}) < d(v_{p_i}, v_{p_{i+1}}) + d(v_{p_{i+1}}, v_{p_{i+2}})$, and replace $v_{p_{i+1}}$ by $v'_{p_{i+1}}$. This process can be repeated for all consecutive links of P until no further improvement is possible. This method can be extended to a three-link look-ahead scheme, too. This is applicable to the iterative process of the aforementioned three heuristics.

4.3. Balanced tree construction

When the mule arrives at subnetwork $\mathcal{G}_i$, m enters this phase to form a data-collection tree rooted at itself to conduct intra-subnetwork data gathering. Specifically, once m lands at the landing port of $\mathcal{G}_i$, m will switch from movement state to data gathering state, request the sensors within m's communication range to be gateway nodes, and instruct all sensors in $\mathcal{G}_i$ to report their sensory data through these gateway nodes to m. Our goal is to balance the loads among these gateway nodes such that the minimum remaining energy among sensors in $\mathcal{G}_i$ is maximized. For this purpose, we extend the centralized algorithm proposed in [41] to a distributed one. We will use the remaining energies of nodes to measure the degree of balance of a tree. The protocol has two stages: *tree construction stage* and *balancing stage*. The first stage is to form an initial tree T in a top–down manner based on the remaining energies of nodes. The second stage is to adjust T in a top–down manner according to nodes' *balanced degrees*.

4.3.1. Tree construction stage.

In this stage, an initial tree will be spanned from m to all nodes of $\mathcal{G}_i$. A *Contact*$(\mathcal{G}_i)$ message will be broadcast by m to its direct neighbors.[§]

(1) When a node v_k receives a *Contact*$(\mathcal{G}_i)$ message, it becomes a gateway node and immediately replies an *Association*(v_k) message to m to become m's child. Then, v_k broadcasts a *Form_Tree*$(e(v_k))$ message to span its subtree, where $e(v_k)$ is v_k's current remaining energy.

(2) When non-gateway node v_j receives a *Form_Tree*$(\cdot)$ message for the first time, it sets a timer Δ_w. After Δ_w expires, from all *Form_Tree*$(\cdot)$ messages it received, v_j chooses an on-tree node v_p with the maximum $e(v_p)$ as its parent by sending an *Association*(v_j) message to v_p. In case of a tie, the one with the least number of neighbors is chosen. Then, v_j sets itself as an on-tree node and broadcasts a *Form_Tree*$(e(v_j))$ message.

(3) After v_j becomes an on-tree node, it sends a $Join(v_j)$ message to m. After m has received $Join(\cdot)$ messages from all nodes in $\mathcal{G}_i$, tree T is formed, and m will broadcast an $Adjustment(T)$ message to instruct all nodes in $\mathcal{G}_i$ to enter the next stage.

4.3.2. Balancing stage.

In this stage, the adjustment will be conducted in a top–down manner. Given T, a node v_k can compute its remaining energy after executing one round of data collection as follows: $r(v_k) = e(v_k) - E_T(v_k)$. To evaluate the balancing degree of T with respect to v_k, we define $B(T, v_k) = \max_{v_j \in C(v_k)}\{r(v_j)\} - \min_{v_j \in C(v_k)}\{r(v_j)\}$, where $C(v_k)$ is the set of v_k's children in T. Note that a smaller $B(\cdot)$ means better balance in terms of remaining energy. The adjustment works as follows:

(1) When v_k receives an $Adjustment(T)$ message from its parent, it computes its current $B(T, v_k)$. Then, let $v_{\min}$ (resp., $v_{\max}$) be the child of v_k, which has the smallest (resp., largest) remaining energy $r(\cdot)$. For each subtree of $v_{\min}$, v_k tries to move it to $v_{\max}$ (if the connectivity exists) and computes the new balancing degree $B(T', v_k)$, where T' is the new tree. If there exists a movement that leads to the smallest balancing degree less than its current one, then v_k instructs $v_{\min}$ and $v_{\max}$ to do so.

(2) If v_k makes any change of T in step 1, it goes back to step 1 and tries another change. Otherwise, it broadcasts an $Adjustment(T)$ message to its children and replies a $Complete(v_k)$ message to m.

(3) After m has collected $Complete(\cdot)$ messages from all nodes in $\mathcal{G}_i$, this phase completes.

4.4. Complexity analysis

In this section, we analyze the time complexity of the proposed heuristics. In Section 5, we will further investigate the performance issue through simulations. Let $N = |V|$ be the number of nodes, $\mathcal{E}$ be the number of communication links, n be the number of original subnetworks, and n' be the number of subnetworks after partitioning. Normally, $N \gg n' \geq n$.

For the subnetwork partition phase, step 1 takes $O(n + n')$ to compute the initial seeds. The time complexity of steps 2–3 is the same as the breadth-first search, that is, $O(N + \mathcal{E})$. Step 4 takes $O(N)$ time to scan all nodes for updating seeds. Suppose that steps 2–4 are repeated I times. Normally, $I \ll N$ in practice because our subnetwork partition phase is extended from the k-means algorithm [43]. Finally, step 6 will repeat the aforementioned process up to $O(n')$ times. Therefore, the total time complexity of the subnetwork partition phase is $O((n + n') + ((N + \mathcal{E}) + N)In') = O((N + \mathcal{E})In')$.

[§]In practice, any localization algorithm will suffer from some degree of localization errors, typically ranged between $0.2r_c$ and r_c [42]. This may lead to the mule missing the landing port or connecting to an incorrect landing port. The mule may need to circle around to locate the landing port or use its neighboring nodes as the landing port. Through simulations, our protocol still works correctly under such situations but may suffer from little performance degradation in the minimum remaining energy for the CH-based scheme. Clearly, if we allow the mule to pick a landing port different from the originally planned one (which is at the center-of-gravity), a larger localization error may cause a less balanced data-collection tree.

For the path-planning phase, there are three schemes. For the greedy scheme, each iteration takes $O(N)$ time to find the next landing port, and there are n' iterations. Thus, its time complexity is $O(Nn')$. For the CH-based scheme, step 1 takes $O(N)$ time to scan all nodes for computing the delegation nodes. Finding a CH in step 2 takes $O(n'r)$ time by Jarvis' march [33], where $r \leq n'$ is the number of nodes along the CH. In step 3, at most $O(n')$ unvisited delegation nodes will be checked in each iteration, and there are $O(n')$ iterations. In addition, all links in P will be checked, giving cost of $O(n'^3)$. Thus, the total complexity is $O(N + n'^3)$. For the CC-based scheme, step 1 costs $O(N)$. In step 2, finding the next node in the CC takes $O(N)$, and there are $O(r)$ iterations. Thus, step 2 takes $O(Nr)$ time. In steps 3–4, each insertion will try $O(N)$ nodes. Thus, steps 3–4 take $O(Nn'^2)$ time. The total complexity is $O(Nn'^2)$.

For the balanced tree construction phase, we first analyze the computational cost and the message complexity of each node. For the computational cost, in the tree construction stage, each node v_k takes $O(N)$ time to check those received $Form\_Tree(\cdot)$ messages for selecting its parent. In the balancing stage, for a pair of $v_{\min}$ and $v_{\max}$, for each possible movement T', each node v_k will take $O(N)$ time to compute a new $B(T', v_k)$ value. Because there are $O(N^2)$ combinations of movements and $O(N^2)$ possible pairs of $v_{\min}$ and $v_{\max}$, the balancing stage takes $O(N^5)$ time. Overall, the balanced tree construction phase takes $O(N^5)$. As to the message complexity, the tree construction stage is similar to forming an MST. Thus, the message complexity of this stage is $O(N)$. In the balancing stage, each node v_k sends at most $O(N^2)$ $Adjustment(T)$ messages to its $v_{\min}$ and $v_{\max}$, and there are $O(N^2)$ pairs of $v_{\min}$ and $v_{\max}$. Therefore, the balancing stage incurs $O(N^4)$ message complexity. Overall, the balanced tree construction phase incurs $O(N^4)$ message complexity.

4.5. Extensions to multiple mobile mules

The aforementioned solutions have assumed that there is only one mule. In the following, we show how to extend EM-TSP to multiple mules.

Definition 4. *Given an SS-WSN $G = (V, E)$, a sink node v_0 in G_0, an energy threshold τ_e, and K mobile mules located at v_0, the min–max EM-TSP is to find a proper partition of G into subnetworks and K traversal paths $P = \{P_1, P_2, \ldots, P_K\}$, each starting and ending at v_0, visiting each subnetwork $G_i \neq G_0$ in one landing port by at least one path, and connecting all nodes in G_i via a data-collection tree T rooted at the landing port, such that $\max_{v_k \in V} E_T(v_k) \leq \tau_e$ and the maximum of these path lengths is minimized.*

Our three-phase heuristics can be directly applied to min–max EM-TSP, except that the path-planning phase needs to be extended to K traversal paths. In the following, we propose two solutions. The first one is to group subnetworks into K *clusters* by applying the traditional k-means algorithm [39] before planning paths. First, the center-of-gravities of all subnetworks are identified. Then, the k-means algorithm is applied to all subnetworks, except $\mathcal{G}_0$, into K clusters. Then, for each cluster of subnetworks, any one of our earlier path-planning schemes is applied.

The k-means algorithm has more sense of the geographic vicinity of subnetworks but little sense of load balance. The second heuristic is to iteratively merge clusters until only K clusters remain. Initially, each subnetwork is regarded as a cluster. For each cluster, we merge $\mathcal{G}_0$ into this cluster and compute a tentative traversal path by any single-mule scheme. If there are sufficient mules (i.e., the number of clusters is less than K), the algorithm stops. Otherwise, we merge the cluster with the shortest traversal path with the cluster nearest to it. The distance between two clusters are defined as the minimum distance between any two nodes between these two clusters. If there are K clusters, then the algorithm terminates. Otherwise, we repeat the aforementioned process to merge more clusters.

5. SIMULATION RESULTS

A simulator has been implemented by JAVA programs. To simulate an SS-WSN, we randomly deployed N sensor nodes in an $S \times S$ m^2 field. The field is divided into grids, each of size $s_g \times s_g$ m^2. In order to form a spatially separated network in a systematic way, we imposed a fail probability of P_f on each grid. If a grid is determined to fail, all sensor nodes inside it fail. This would partition the network into multiple subnetworks when P_f is sufficiently large. The sink node v_0 is randomly selected. Following the energy model of Mica2 [44], we set $e = 100$ mJ. Considering long-term monitoring applications [45], we set $\lambda = 10$ packets/h, with 10 kb per packet. Table I summarizes all default

Table I. Definitions of parameters and their default values used in our simulations.

Parameter	Meanings	Default value
N	Number of sensor nodes	1000
S	Area size	500 m
s_g	Grid size	50 m
P_f	Fail probability of a grid	0.5
r_c	Transmission range of a node	20 m
τ_e	Energy consumption threshold	3×10^6 mJ
e	Energy cost to transmit one unit of data	100 mJ
λ	Data arrival rate	10 packets/h
δ	Maximum duration between rounds	100 h
E_{total}	Initial energy of a node	10^7 mJ
K	Number of mobile mules	1

parameters used in our simulations. We compared our EM-TSP solutions against a *modified Christofides heuristic* in which the path-planning phase randomly chooses a landing port for each subnetwork after partitioning and then the Christofides heuristic is applied to find a traversal path of the mule. All simulation results are from the average of 100 runs.

We consider two performance metrics: (i) the path length of a mule and (ii) the minimum remaining energy among sensors. In the following, we first vary the number of sensor nodes N, the area size S, the fail probability P_f, and the transmission range r_c, to investigate the performance when there is only one mobile mule. In addition, we also simulate a more complicated irregular radio propagation model. Finally, we compare our heuristics against an exhaustive search by varying the number of subnetworks before partitions. Then, we study the performance when multiple mules coexist.

5.1. Effect of *N*

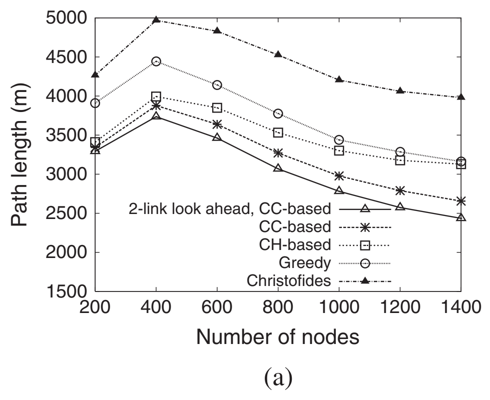
First, we investigate the effect of the number of nodes. Figure 9(a) shows its impact on path length. The CC-based scheme performs the best, which implies that its local optimization technique can efficiently shorten the path length. When N falls in the range 200–400, the path length is relatively longer. This is because there are too many (original) subnetworks, as reflected by Figure 9(b). As N increases ($N = 400 \sim 1400$), the path length decreases gradually because there are less subnetworks to be visited. Recall that the ideal number of partitions is $\sum_{i=0}^{n} \lceil \frac{|G_i|}{\tau_s} \rceil$. Figure 9(b) shows that our schemes will not incur too many subnetworks after partitioning. Figure 10 shows that the CH-based scheme performs the best in terms of the minimum remaining energy of nodes. This is because the CH-based scheme selects the node close to the geometrical centroid of each subnetwork as the landing port. Thus, the average relaying hop counts from sensors to the gateway nodes are reduced. As can be seen, although some of our path-planning policies choose landing ports nearby the boundaries of subnetworks, the resulted minimum remaining energy among sensors is still slightly less than that of Christofides. Additionally, we can see that the greedy and CC-based schemes have the similar performance in terms of the minimum remaining energy among sensor nodes. This is because these schemes will choose nodes around the borders of subnetworks as landing ports. Thus, the average amount of sensory data relayed by a sensor becomes higher than that in the CH-based scheme. When N is relatively large, the minimum remaining energy is relatively small and eventually becomes flat. This is because the energy consumption of each sensor can be bounded by τ_e even if there is the larger N.

5.2. Effect of *S*

Next, we study the effect of the area size by varying S from 400 to 800. Figure 11(a) shows that the mule's traversal path length will increase proportionally with S. This is caused by two reasons: (i) there are more and more small subnetworks as the network is becoming sparser, as reflected by Figure 11(b), and (ii) the distance between subnetworks are relatively farther. Even when S is large (which means that subnetworks have less nodes), our solutions still outperform Christofides. This gives us evidence that the selection of landing ports is important and the existing TSP heuristics (such as Christofides) cannot be trivially applied to EM-TSP to achieve good performance. In Figure 12, the minimum remaining energy of sensor nodes increases as S increases. This is because the larger S will cause that each subnetwork includes the less sensor nodes and spends less energy on relaying.

5.3. Effect of P_f

We now investigate the effect of the fail probability by varying P_f from 0.9 to 0.1. As shown in Figure 13(a), the CC-based scheme still performs better than the other schemes in terms of path length. The gap between the CC-based scheme and others actually enlarges as P_f decreases.



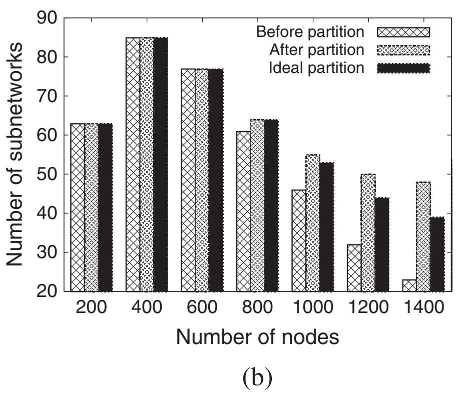


Figure 9. Effect of $N = 200 \sim 1400$ on (a) path length and (b) number of subnetworks.

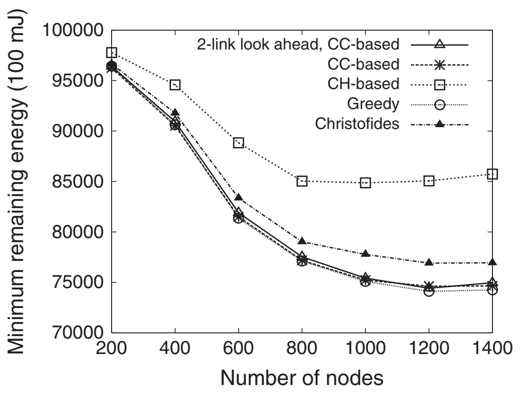


Figure 10. Effect of $N = 200 \sim 1400$ on the minimum remaining energy of sensor nodes.

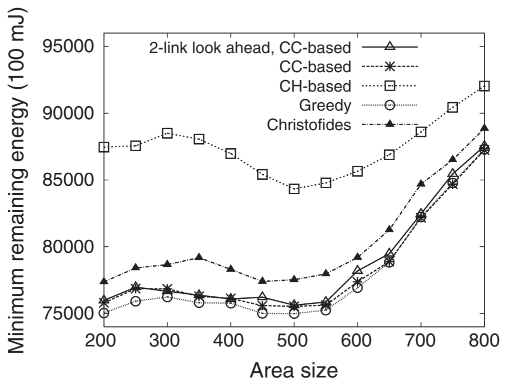


Figure 12. Effect of $S = 400 \sim 800$ on the minimum remaining energy of sensor nodes.

This is because a smaller P_f will lead to more available sensors and, thus, larger-scale (original) subnetworks and more (logical) subnetworks. This also concludes that the CC-based is more important when we deploy larger-scale subnetworks. Note that in Figure 13(b), it is reasonable to see that the path length is related to the number of subnetworks after partition (rather than that before partition). From Figure 13(b), we see that the gaps between the number of subnetworks before partition and the ideal number of subnetworks are quite large when P_f is relatively small. This is due to two reasons. First, it is possible to have isolated nodes when nodes are randomly deployed. Second, the connectivity of nodes nearby the boundary of the field is not strong compared with those inner nodes. In Figure 14, the minimum remaining energy among sensors decreases as P_f decreases because sensors spend more energy on relaying.

5.4. Effect of r_c

We vary r_c from 20 to 50 to study the effect of nodes' transmission range. Figure 15(a) shows that the path length decreases as r_c increases because sensor nodes will connect each other easily. Figure 15(b) also gives evidence that most subnetworks have many sensor nodes and need to be partitioned when a larger r_c is considered. Figure 16 shows that the minimum remaining energy of sensors will be bounded by τ_e even if the larger r_c causes that each subnetwork includes more sensors. By the aforementioned simulation results, we conclude that our schemes can efficiently solve the bi-objective EM-TSP.

5.5. Effect of irregular radio propagation

To understand the impact of more complicated radio propagation, we adopt the degree of irregularity (DOI) model in [42,46], which allows us to vary the radio range in different directions. For example, when $DOI = 0.1$, the radio range is randomly chosen from $[0.9r_c, 1.1r_c]$ in each direction. We vary DOI from 0.1 to 0.5 in the following simulations. Figure 17(a) shows that the path length of the mule decreases as DOI increases. This is because a sensor node may discover more neighbor nodes, leading to more opportunities to reduce the number of subnetworks. Figure 17(b)

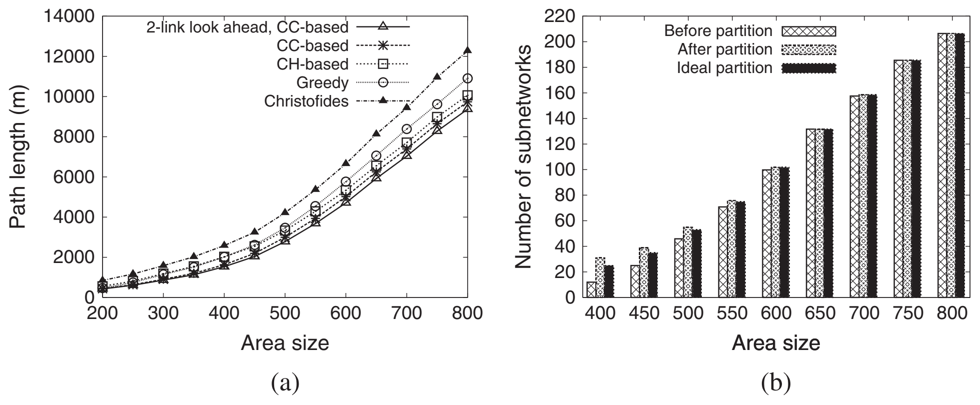


Figure 11. Effect of $S = 400 \sim 800$ on (a) path length and (b) number of subnetworks.

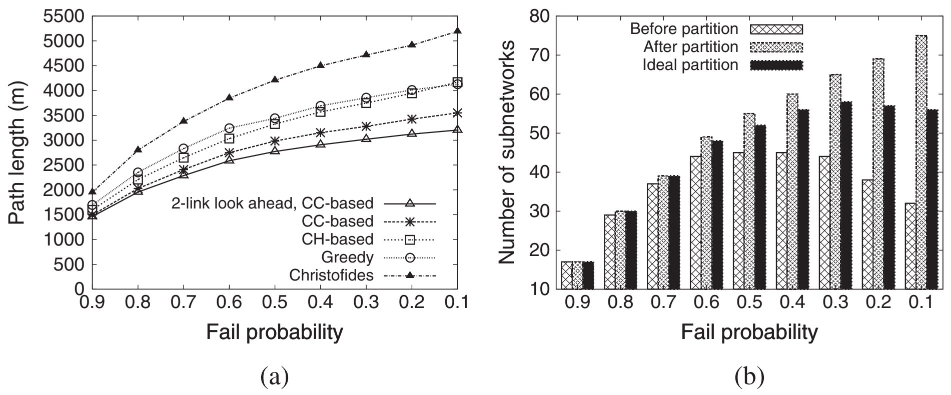


Figure 13. Effect of $P_f = 0.9 \sim 0.1$ on (a) path length and (b) number of subnetworks.

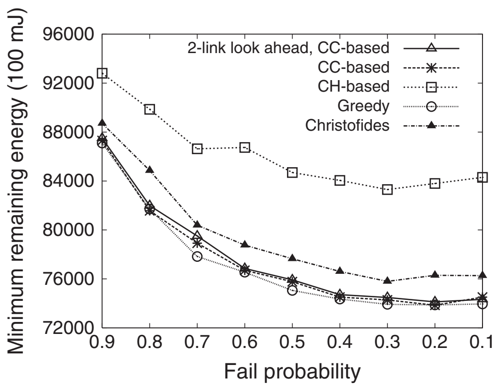


Figure 14. Effect of $P_f = 0.9 \sim 0.1$ on the minimum remaining energy of sensor nodes.

gives evidences of this argument. Figure 18 shows that the irregularity of radio propagation does not have a serious impact on energy consumption of sensors because it only changes the topology slightly.

5.6. Comparison to exhaustive search

To understand how our path-planning heuristics perform against an exhaustive search algorithm, which is able to find the optimal path, we conducted the following simulations. We vary the value of n but skipped the subnetwork partition step in Section 4.1 (this is to ensure that our observation is not affected by the partitioning result). Unfortunately, only a small n (around 10) can be computationally handled by an exhaustive scheme (note that a subnetwork may contain a lot of sensor nodes, which also need be searched.) Figure 19 shows that the CC-based schemes perform very closely to the optimal scheme.

5.7. Effect of *K*

Finally, we consider the case of multiple coexisting mules. We look at the maximum path length among all mules. First, we vary K from 1 to 20 to investigate the effect of the number of mules when the merging-based clustering scheme is adopted. Figure 20(a) shows that, as the number of mules increases, the gaps between different path-planning schemes shrink gradually. This is because each

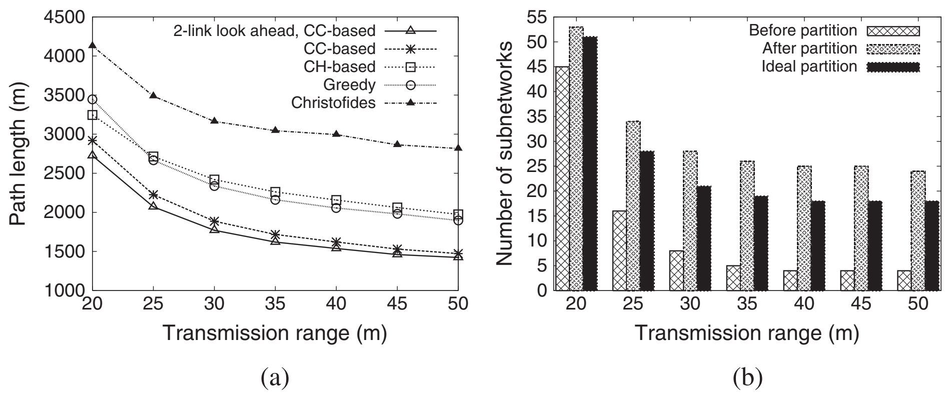


Figure 15. Effect of $r_c = 20 \sim 50$ on (a) path length and (b) number of subnetworks.

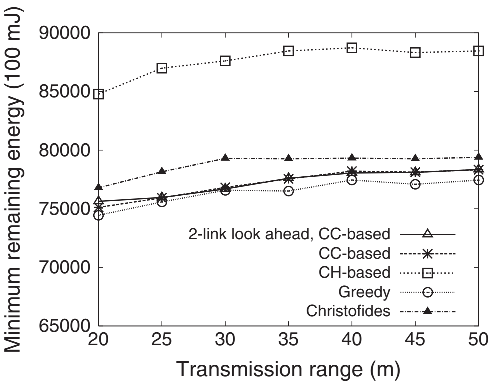


Figure 16. Effect of $r_c = 20 \sim 50$ on the minimum remaining energy of sensor nodes.

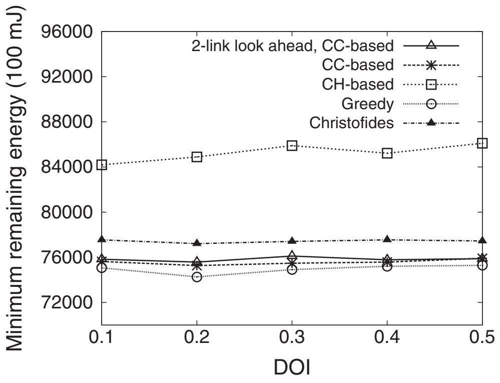


Figure 18. Effect of $DOI = 0.1 \sim 0.5$ on minimum remaining energy of sensor nodes.

mule is responsible for traversing relatively less subnetworks. It is to be noted that, ideally, K mules should be able to reduce the path lengths by $\frac{1}{K}$. As can be seen in the figure, the reduction in the maximum path length is much lesser than what was expected. This shows the importance of load balance among mules. In Figure 20(b), we compare the performance of the merging-based and k-means-based schemes, where the CC-based scheme is adopted. Generally speaking, the k-means-based scheme performs better. As K increases, the gaps decrease because each mule needs to take care of relatively less subnetworks.

To summarize, the greedy scheme has low computation cost, the CH-based scheme incurs low energy costs on sensor nodes, and the CC-based scheme can find shorter data-collection paths. Therefore, these three path-planning schemes may be adopted in different scenarios. For example, the greedy scheme is more suitable for a network with a high failure rate because frequent recomputing of the mule's path may be needed. The CH-based scheme is more suitable for a sparse network because energy of sensor nodes becomes more critical for connectivity reason. On the other hand, the CC-based scheme is more suitable for a delay-sensitive network with a stronger real-time requirement.

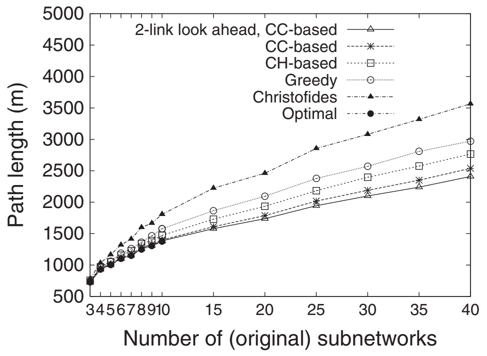


Figure 19. Comparing with the optimal solution on path length.

6. CONCLUSIONS

This paper considers the data gathering issue in an SS-WSN, where sensor nodes may form several isolated subnetworks, each far away from each other. Mobile mules are adopted to traverse these subnetworks to conduct data collection. To address issues of data-collection latency

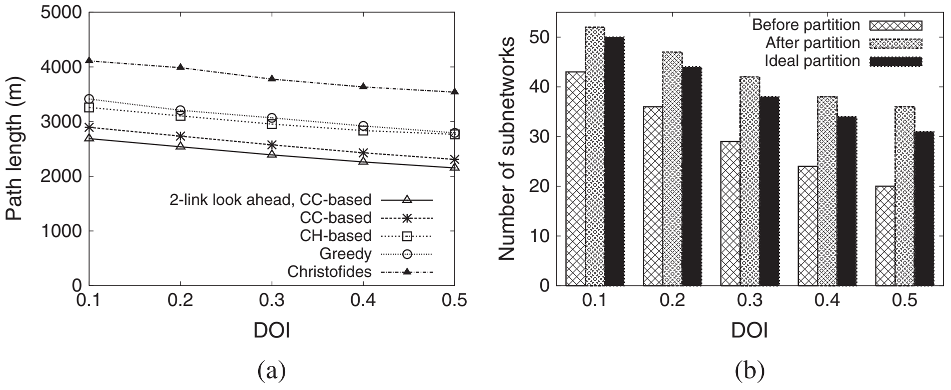
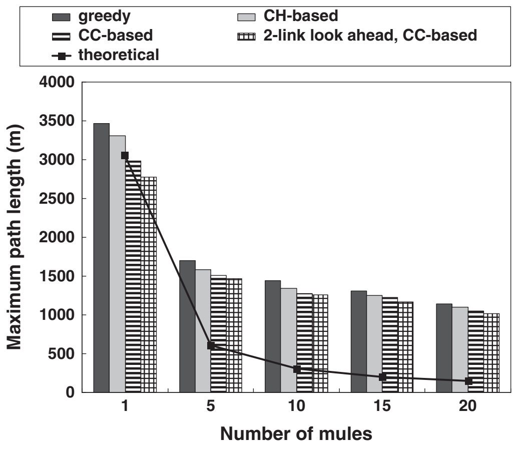
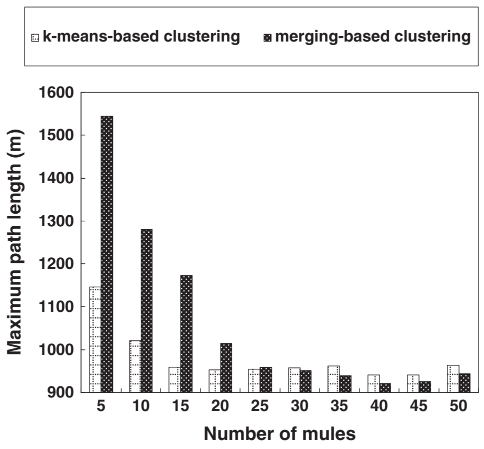


Figure 17. Effect of $DOI = 0.1 \sim 0.5$ on (a) path length and (b) number of subnetworks.



(a)



(b)

Figure 20. Effect of the number of mules on maximum path length among mules, where (a) different path-planning schemes and (b) different clustering schemes.

and network lifetime simultaneously, we formulate a new problem, called EM-TSP, to find mules' traversal paths to visit each subnetwork in at least one landing port such that the energy consumption of sensors is bounded and the traversal path lengths of mules are minimized. We show that EM-TSP is a generalization of the classical TSP. Based on some interesting geometrical properties, we propose several heuristics to solve EM-TSP. In particular, the properties of the CH are explored to solve this problem. Simulation results show that our approaches can find efficient solutions to EM-TSP so as to balance between data-collection latency and network lifetime.

APPENDIX A: PROOF OF THEOREM 1

Proof. We prove that LEM-TSP is NP-hard by reducing the ETSP [7], an NP-hard problem, to a special case of LEM-TSP. Note that ETSP is a special case of the TSP when nodes are given in a Euclidean space. Let $G' = (V', E')$ and a positive integer L' be an arbitrary instance of ETSP, where a complete graph and the distance between any two vertices v'_i and v'_j in V' is defined as their Euclidean distance. ETSP is to determine whether G' has a tour P' visiting all vertices in V' such that $|P'| \leq L'$. We can reduce $G' = (V', E')$ to a special case $G = (V, E)$ of LEM-TSP in polynomial time as follows. Let $V = V'$ and $E = \emptyset$ (i.e., the communication range of each sensor node is infinitesimal, and thus, each subnetwork G_i, $i = 0, 1, \ldots, |V - 1|$, contains only one node). The distance between any two vertices v_i and v_j in V is also their Euclidean distance. The sink node v_0 can be any vertex in V. Let the energy threshold $\tau_e = e \cdot \lambda \cdot \delta$ and $L = L'$. Clearly, the reduction can be done in polynomial time.

We now show that G' has a tour P' visiting all vertices in V' such that $|P'| \leq L'$ iff G has a partition and a traversal path P starting and ending at v_0, visiting each $G_i \neq G_0$ in one landing port, and connecting all nodes in G_i via a data-collection tree T rooted at the landing port such that $\max_{v_k \in V} E_T(v_k) \leq \tau_e$ and $|P| \leq L$. We first prove the *if* part. If G has a solution to LEM-TSP with a traversal path P and $|P| \leq L$, P must visit each vertex in G exactly once because the only way to partition G is to make that each G_i includes only one node v_i. Thus, we can find a corresponding path P' in G' with the same visiting order of vertices as P such that $|P'| \leq L'$. Conversely, we then prove the *only if* part. If G' has a solution P' to ETSP with $|P'| \leq L'$, we can find a corresponding path P in G with the same visiting order of vertices as P' and a partition of G that each subnetwork G_i contains only one node v_i. Clearly, P starts and ends at v_0, visits each subnetwork G_i in one landing port v_i, and connects all nodes in G_i via a data-collection tree T (i.e., T is a single-node tree) such that $\max_{v_k \in V} E_T(v_k) \leq \tau_e$ and $|P| \leq L$. □

ACKNOWLEDGEMENTS

Yu-Chee Tseng's research is co-sponsored by MoE ATU Plan; by NSC grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, 98-2219-E-009-005, and 99-2218-E-009-005; by ITRI, Taiwan; by III, Taiwan; by D-Link; and by Intel.

REFERENCES

1. Rapaka A, Madria S. Two energy efficient algorithms for tracking objects in a sensor network. *Wireless Communications and Mobile Computing* 2007; **7**(6): 809–819.
2. Hu F, Xiao Y, Hao Q. Congestion-aware, loss-resilient bio-monitoring sensor networking for mobile health

applications. *IEEE Journal Selected Areas in Communications* 2009; **27**(4): 450–465.

3. Liu H, Wan P, Jia X. Maximal lifetime scheduling for sensor surveillance systems with k sensors to one target. *IEEE Transactions on Parallel and Distributed Systems* 2006; **17**(12): 1526–1536.
4. Tubaishat M, Zhuang P, Qi Q, Shang Y. Wireless sensor networks in intelligent transportation systems. *Wireless Communications and Mobile Computing* 2009; **9**(3): 287–302.
5. Terrestrial Ecology Observing Systems, Center For Embedded Networked Sensing. http://research.cens.ucla.edu/.
6. Data Mule, Center For Embedded Networked Sensing. http://research.cens.ucla.edu/projects/2005/Actuation/datamule/.
7. Papadimitriou CH. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science* 1977; **4**: 237–244.
8. Golden B, Bodin L, Doyle T, Stewart W, Jr. Approximate traveling salesman algorithms. *Operations Research* 1980; **28**(3): 694–711.
9. Wang Y-C, Wu F-J, Tseng Y-C. Mobility management algorithms and applications for mobile sensor networks. *Wireless Communications and Mobile Computing*. DOI: 10.1002/wcm.886.
10. Zou Y, Chakrabarty K. Sensor deployment and target localization in distributed sensor networks. *ACM Transactions on Embedded Computing Systems* 2004; **3**(1): 61–91.
11. Heo N, Varshney PK. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans* 2005; **35**(1): 78–92.
12. Wang G, Cao G, La Porta TF. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing* 2006; **5**(6): 640–652.
13. Akkaya K, Younis M. COLA: a coverage and latency aware actor placement for wireless sensor and actor networks. In *Proceedings of IEEE Vehicular Technology Conference*, 2006; 1–5.
14. Koc O, Jaikaeo C, Shen C-C. Navigating actors in mobile sensor actor networks. In *Proceedings of the ACM Workshop on Sensor and Actor Networks*, 2007; 19–26.
15. Akkaya K, Senel F. Detecting and connecting disjoint sub-networks in wireless sensor and actor networks. *Ad Hoc Networks* 2009; **7**(7): 1330–1346.
16. Yang G, Tong B, Qiao D, Zhang W. Sensor-aided overlay deployment and relocation for vast-scale sensor networks. In *Proceedings of IEEE INFOCOM*, 2008; 2216–2224.
17. Shah RC, Roy S, Jain S, Brunette W. Data MULEs: modeling a three-tier architecture for sparse sensor networks. In *Procedings of IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003; 30–41.
18. Jain S, Shah RC, Brunette W, Borriello G, Roy S. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mobile Networks and Applications* 2006; **11**(3): 327–339.
19. Luo L, Huang C, Abdelzaher T, Stankovic J. EnviroStore: a cooperative storage system for disconnected operation in sensor networks. In *Proceedings of IEEE INFOCOM*, 2007; 1802–1810.
20. Singh A, Krause A, Kaiser WJ. Nonmyopic adaptive informative path planning for multiple robots. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009; 1843–1850.
21. Zhao W, Ammar MH. Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks. In *Proceedings of the IEEE Workshop Future Trends in Distributed Computing Systems*, 2003; 308–314.
22. Zhao W, Ammar M, Zegura E. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2004; 187–198.
23. Zhao W, Ammar M, Zegura E. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proceedings IEEE INFOCOM*, 2005; 1407–1418.
24. Ngai EC-H, Liu J, Lyu MR. Delay-minimized route design for wireless sensor-actuator networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, 2007; 3675–3680.
25. Ngai EC-H, Liu J, Lyu MR. An adaptive delay-minimized route design for wireless sensor–actuator networks. *IEEE Transactions on Vehicular Technology* 2009; **58**(9): 5083–5094.
26. Yuan B, Orlowska M, Sadiq S. On the optimal robot routing problem in wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering* 2007; **19**(9): 1252–1261.
27. Wan C-Y, Eisenman SB, Campbell AT, Crowcroft J. Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of the ACM International Conference Embedded on Networked Sensor Systems*, 2005; 116–129.
28. Ma M, Yang Y. Data gathering in wireless sensor networks with mobile collectors. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, 2008; 1–9.
29. Xing G, Wang T, Xie Z, Jia W. Rendezvous planning in wireless sensor networks with mobile elements. *IEEE Transactions on Mobile Computing* 2008; **7**(12): 1430–1443.

30. Xing G, Wang T, Jia W, Li M. Rendezvous design algorithms for wireless sensor networks with a mobile base station. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2008; 231–240.
31. Rao J, Wu T, Biswas S. Network-assisted sink navigation protocols for data harvesting in sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2008; 2887–2892.
32. Rao J, Biswas S. Joint routing and navigation protocols for data harvesting in sensor networks. In *Proceedings of the IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, 2008; 143–152.
33. Cormen TH. *Introduction to Algorithms*. The MIT Press: US, 2001.
34. Christofides N. Worst-case analysis of a new heuristic for the travelling salesman problem. *Report No. 388*, GSIA, Carnegie-Mellon University, Pittsburgh, PA, 1976.
35. Arora S. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM* 1998; **45**(5): 753–782.
36. Meliou A, Chu D, Guestrin C, Hellerstein J, Hong W. Data gathering tours in sensor networks. In *Proceedings of the IEEE International Symposium on Information Processing in sensor networks*, 2006; 43–50.
37. Meliou A, Krause A, Guestrin C, Hellerstein J. Non-myopic informative path planning in spatio-temporal models. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, 2007; 602–607.
38. Keshavarzian A, Lee H, Venkatraman L. Wakeup scheduling in wireless sensor networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2006; 322–333.
39. Hartigan JA. *Clustering Algorithms*. Wiley: New York, 1975.
40. Larson RC, Odoni AR. *Urban Operations Research*. Prentice-Hall: Englewood Cliffs, NJ, 1981.
41. Dai H, Han R. A node-centric load balancing algorithm for wireless sensor networks. In *Proceedings of the IEEE Global Telecommunications Conference*, 2003; 548–552.
42. Hu L, Evans D. Localization for mobile sensor networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking*, 2004; 45–57.
43. Duda RO, Stork DG, Hart PE. *Pattern Classification*. John Wiley and Sons, Inc.: New York, 2000.
44. Shnayder V, Hempstead M, Chen B-R, Allen GW, Welsh M. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems*, 2004; 188–200.
45. Hartung C, Han R, Seielstad C, Holbrook S. FireWxNet: a multitiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services*, 2006; 28–41.
46. Zhou G, He T, Krishnamurthy S, Stankovic JA. Models and solutions for radio irregularity in wireless sensor networks. *ACM Transactios on Sensor Networks* 2006; **2**(2): 221–262.

AUTHORS' BIOGRAPHIES

Fang-Jing Wu received the B.S. degree in Mathematics form the Fu Jen Catholic University and the M.S. degree in Computer Science and Information Engineering from the National Chiao-Tung University, Taiwan, in 2001 and 2004, respectively. She was a research assistant in the Department of Communication Engineering, National Chiao-Tung University, Taiwan, in 2004. She is currently pursuing Ph.D. in the Department of Computer Science, National Chiao-Tung University, Taiwan. Her current research interests are primarily in pervasive computing and wireless sensor networks.

Yu-Chee Tseng got his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He is/was Professor (2000–present), Chairman (2005–2009), and Associate Dean (2007–2011) of the Department of Computer Science, National Chiao-Tung University, Taiwan, and Chair Professor, Chung Yuan Christian University (2006–2010).

Dr. Tseng received Outstanding Research Award (National Science Council, 2001, 2003, and 2009), Best Paper Award (Int'l Conf. on Parallel Processing, 2003), Elite I. T. Award (2004), and Distinguished Alumnus Award (Ohio State University, 2005), and Y. Z. Hsu Scientific Paper Award (2009). His research interests include mobile computing, wireless communication, and parallel and distributed computing.

Dr. Tseng serves/served on the editorial boards of *IEEE Transactions on Vehicular Technology* (2005–2009), *IEEE Transactions on Mobile Computing* (2006–present), and *IEEE Transations on Parallel and Distributed Systems* (2008–present).