Mobility Management Algorithms and Applications for Mobile Sensor Networks

You-Chiun Wang, Fang-Jing Wu, and Yu-Chee Tseng

Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan
Email: {wangyc, fangjing, yctseng}@cs.nctu.edu.tw

Abstract

Wireless sensor networks (WSNs) offer a convenient way to monitor physical environments. In the past, WSNs are all considered static to continuously collect information from the environment. Today, by introducing intentional mobility to WSNs, we can further improve the network capability on many aspects, such as automatic node deployment, flexible topology adjustment, and rapid event reaction. In this article, we survey recent progress in mobile WSNs and compare works in this field in terms of their models and mobility management methodologies. The discussion includes three aspects. Firstly, we discuss mobility management of mobile sensors for the purposes of forming a better WSN, enhancing network coverage and connectivity, and relocating some sensors. Secondly, we introduce path-planning methods for data ferries to relay data between isolated sensors and to extend a WSN's lifetime. Finally, we review some existing platforms and discuss several interesting applications of mobile WSNs.

KEY WORDS: mobility management, path planning, sensor applications, topology adjustment, wireless sensor networks.

1. Introduction

The development of wireless technologies and micro-sensing MEMS has triggered the success of *wireless sensor networks (WSNs)*. A WSN is composed of one or multiple remote sinks and many tiny, low-power sensors, each equipped with actuators, sensing devices, and wireless transceivers [1]. These sensors are massively deployed in a *region of interest (ROI)* to continuously collect and report surrounding data. WSNs offer a convenient way to monitor physical environments. Many applications such as object tracking, health monitoring, security surveillance, and intelligent transportation [2, 3, 4, 5] have been proposed.

A WSN is usually deployed with static sensors to perform monitoring missions. However, due to the dynamics of events or environments, a purely static WSN could face these challenges: (1) Sensors are often scattered in a ROI by aircrafts or robots [6]. These randomly scattered sensors could not guarantee complete coverage of the ROI and may be partitioned into disconnected subnetworks. The existence of obstacles could even worsen the problem. (2) Sensors are usually powered by batteries. As some sensors exhaust their energy, holes could appear and the network could be broken. However, in many scenarios, it is quite difficult to recharge sensors or redeploy nodes. (3) A WSN may need to support multiple missions or have multiple types of sensors [7]. Sometimes, we may need to send a certain type of sensors to particular locations to support particular needs. Without mobility, this is difficult to achieve. (4) While most efforts assume that sensors are cheap, some types of sensors may be expensive. Dispatching of those expensive ones from locations to locations may be necessary.

By introducing mobility to a WSN, we can enhance its capability to handle the above problems. Nevertheless, mobile WSN and *mobile ad hoc network (MANET)* are essentially different. Mobility in a MANET is often arbitrary, whereas mobility in a mobile WSN should be ‘intentional’, in the sense that we can control their movement to achieve our missions. In this article, we give a comprehensive survey of recent progress in mobile WSNs. Our discussion focus on two types of nodes: *mobile sensors* and *data ferries*. With the former, one may change the network topology by moving these mobile sensors. With the latter, one may maneuver these data ferries to collect or relay sensing data. We will cover three topics:

- *Mobility management of mobile sensors:* First, we introduce deployment methods to organize a WSN. Second, we present relocation methods to improve the coverage and connectivity of a WSN. Third, we discuss how to assign mobile sensors to desired locations.
- *Path planning of data ferries:* We first introduce path-planning methods to maneuver data ferries in a sparse WSN, and then discuss how to use data ferries to extend a WSN’s lifetime.
- *Platforms and Applications of mobile WSNs:* Some mobile platforms and applications will be introduced.

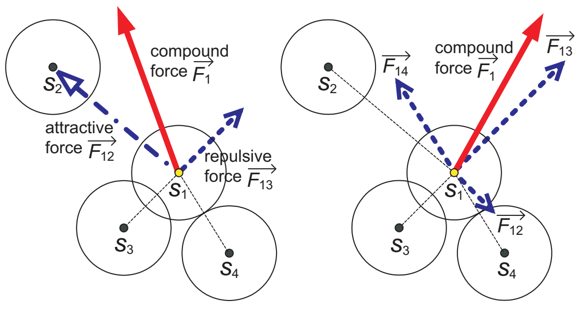
2. Mobility Management of Mobile Sensors

2.1. Solutions to Deploying Mobile Sensors

Sensor deployment is a basic issue since it decides a WSN’s detection ability. A good deployment should satisfy both *coverage* and *connectivity* [8, 9]. Coverage requires that each location in the ROI be monitored by sensors, and connectivity requires that the network remain not partitioned. With mobile sensors, the deployment job becomes ‘automatic’. We introduce three deployment methods: The *force-based deployment* images that virtual forces will drive sensors to move. The *graph-based deployment* identifies uncovered holes and moves sensors to cover them. The *assignment-based deployment* computes the locations to be placed with sensors and then dispatches them in an energy-efficient way.

2.1.1. Force-Based Deployment

The work [10] considers moving sensors by virtual forces. Each sensor s_i is exerted by a compound force $\overrightarrow{F_i} = \overrightarrow{F_{iA}} + \overrightarrow{F_{iR}} + \sum_{j=1,j\neq i}^{n} \overrightarrow{F_{ij}}$, where $\overrightarrow{F_{iA}}$ is an attractive force by the ROI, $\overrightarrow{F_{iR}}$ is the overall repulsive force by obstacles, $\overrightarrow{F_{ij}}$ is the force produced by sensor s_j, and n is the total number of sensors. Each force $\overrightarrow{F_{ij}}$ is denoted by (r_{ij}, θ_{ij}) in a polar coordinate, where r_{ij} is the magnitude and θ_{ij} is the orientation. $\overrightarrow{F_{ij}}$ is expressed as



compound force $\overrightarrow{F_1}$
s_2
attractive force $\overrightarrow{F_{12}}$
repulsive force $\overrightarrow{F_{13}}$
s_1
s_3
s_4
(a) hybrid forces
s_2
$\overrightarrow{F_{14}}$
compound force $\overrightarrow{F_1}$
$\overrightarrow{F_{13}}$
s_1
$\overrightarrow{F_{12}}$
s_3
s_4
(b) repulsive forces

Fig. 1. Examples of force-based deployment.

$$\overrightarrow{F_{ij}} = \begin{cases} (w_A \cdot (d_{ij} - d_{\text{th}}), \theta_{ij}) & \text{if } d_{ij} > d_{\text{th}} \\ (w_R \cdot \frac{1}{d_{ij}}, \pi + \theta_{ij}) & \text{if } d_{ij} < d_{\text{th}} \\ 0 & \text{otherwise,} \end{cases}$$

where w_A/w_R is the measure of an attractive/repulsive force, d_{ij} is the distance between s_i and s_j, and d_{th} is a threshold distance to decide the force type. Fig. 1(a) gives an example, where $d_{\text{th}} = d_{14}$. We see that s_2 exerts an attractive force $\overrightarrow{F_{12}}$, s_3 exerts a repulsive force $\overrightarrow{F_{13}}$, and s_4 exerts no force on s_1 because $d_{12} > d_{th}$, $d_{13} < d_{th}$, and $d_{14} = d_{th}$, respectively. Sensor s_1 is thus moved by the compound force $\overrightarrow{F_1}$.

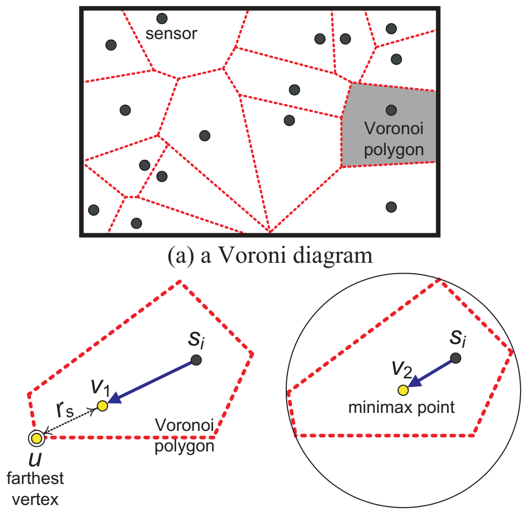
In [11], each sensor is viewed as an electron and is repulsed by other sensors. The force from a higher sensor density area is greater than that from a lower density area, and the force from a nearer sensor is greater than that from a farther sensor. Specifically, the force function $F(\cdot)$ should satisfy three rules: (1) $F(d_{ij}) \geq F(d_{ik})$ if $d_{ij} \leq d_{ik}$. (2) $F(0^+) = F_{\max}$. This gives an upper bound on forces. (3) $F(d_{ij}) = 0$ if $d_{ij} > r_c$ (communication distance). This means that only neighboring sensors will generate forces.

Sensors are moved step by step. In each step, the repulsive force on sensor s_i exerted by a neighboring sensor s_j is $\overrightarrow{F_{ij}} = \frac{D_i}{\mu^2}(r_c |p_i - p_j|) \frac{p_j - p_i}{|p_j - p_i|}$, where D_i is the local sensor density seen by s_i, μ is the expected sensor density after the final deployment, and p_i/p_j is the position of s_i/s_j. The expected sensor density is computed by $\mu = \frac{n \cdot \pi r_c^2}{|\mathcal{A}|}$, where $|\mathcal{A}|$ is the ROI’s area.

Fig. 1(b) shows an example, where s_2, s_3, and s_4 all exert repulsive forces on s_1.

In the above two methods, *oscillation check* and *stability check* are performed to examine whether a sensor has reached its final destination. When a sensor s_i moves back and forth inside a small region many times, it has entered the oscillation state. On the other hand, when s_i moves less than a threshold distance in a fixed duration, it has entered the stable state. In both cases, s_i will stop moving.

Reference [12] considers that sensors work under a probability sensing model. The goal is to deploy a minimum number of sensors such that the detection probability of the ROI is above a predefined threshold. To achieve this, we can first deploy sufficient sensors to satisfy the detection probability. Then, sensors can exert repulsive forces on each other. In this way, the number of sensors may be reduced since some sensors may be pushed outside the ROI.



(a) a Voroni diagram

(b) VOR method (c) Minimax method

Fig. 2. The VOR and Minimax methods.

2.1.2. Graph-Based Deployment

The work [13] adopts a Voronoi diagram to search uncovered holes and moves sensors to cover these holes. Given a set of sensors on a 2D plane, the *Voronoi diagram* [14] consists of a number of *Voronoi polygons* such that each polygon contains one sensor and the points in the polygon are closer to the interior sensor than to other exterior sensors. When the sensing range of a sensor cannot completely cover its Voronoi polygon, there could be an uncovered hole in that polygon. In [13], it proposes the following methods to cover this hole:

Voronoi-based (VOR) method: A sensor should move toward the farthest vertex of its current polygon. Fig. 2(b) gives an example, where the dotted polygon is sensor s_i's current polygon and u is the farthest vertex. Sensor s_i will move along the direction $\overrightarrow{s_i u}$ and stop at v_1, where $|\overline{uv_1}| = r_s$.

Minimax method: A sensor should move to the *minimax point* of its current polygon, where the minimax point of a polygon is the center of the circle with the minimum radius that can cover the whole polygon (refer to [13] for details about finding the circle). Fig. 2(c) gives an example, where v_2 is the minimax point of s_i's current polygon.

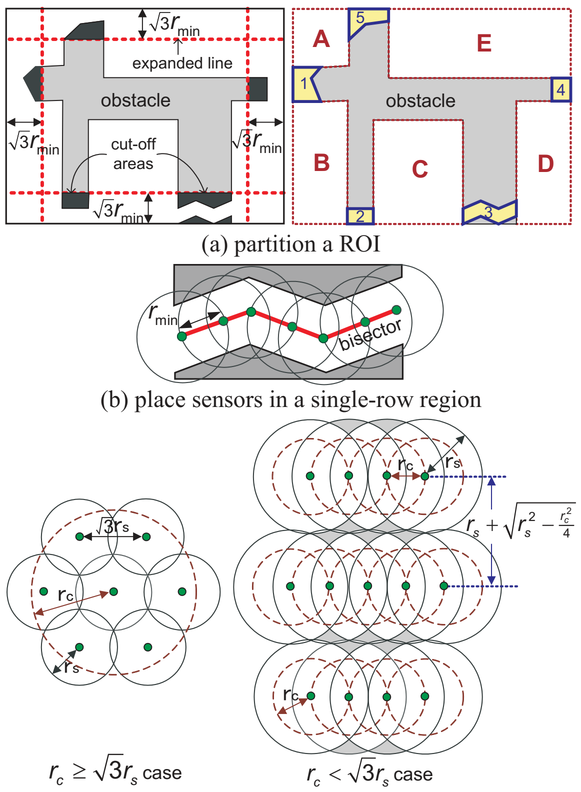
2.1.3. Assignment-Based Deployment

Reference [15] focuses on deployment in ROIs with obstacles. It considers two related problems: *sensor placement* and *sensor dispatch*. The former asks how to use the minimum number of sensors in a ROI to guarantee coverage and connectivity. The latter asks how to dispatch mobile sensors to the designated locations computed by the placement result such that their moving energy is minimized.

To solve the placement problem, [15] partitions a ROI $\mathcal{A}$ into *single-row* and *multi-row* regions. A single-row region requires one row of sensors to cover it, and a multi-row region requires multiple rows of sensors to cover it. To partition $\mathcal{A}$, we first identify all single-row regions, which is achieved by expanding $\mathcal{A}$'s boundaries inward and obstacles' perimeters outward by a distance of $\sqrt{3}r_{\min}$, where $r_{\min} = \min\{r_c, r_s\}$. When the expanded line cuts off an obstacle with an area, we take a project from that area to identify a single-row region. Fig. 3(a) gives an example, where 5 single-row regions (with numbers) are identified. Other regions will be multi-row ones. Then, we place sensors in each region as follows:

Single-row region: We place a sequence of sensors along the region's bisector, each separated by a distance of $r_{\min}$. Fig. 3(b) gives an example.

Multi-row region: Two cases are considered, as Fig. 3(c) shows. When $r_c \geq \sqrt{3}r_s$, adjacent sensors are regularly separated by a distance of $\sqrt{3}r_s$. When $r_c < \sqrt{3}r_s$, sensors in each row are separated by a distance of r_c. Adjacent rows are separated by a distance of $r_s + \sqrt{r_s^2 - \frac{r_c^2}{4}}$ and shifted by a distance of $\frac{r_c}{2}$. To connect adjacent rows, we add a column of sensors between them, each separated by a distance not larger than r_c.



(a) partition a ROI

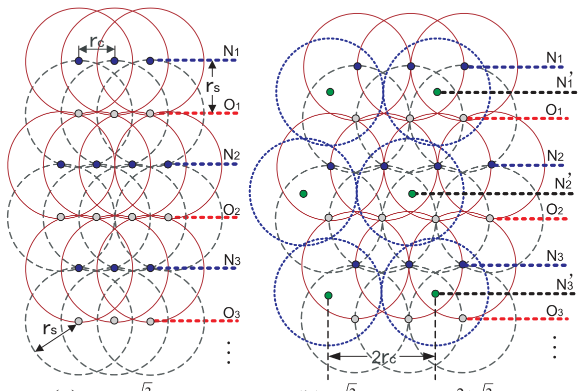
(b) place sensors in a single-row region

(c) place sensors in a multi-row region

Fig. 3. The sensor placement solution proposed in [15].

Given a set of mobile sensors $\mathcal{S}$ and a set of locations $\mathcal{L}$ computed by the placement result, [15] considers dispatching $\mathcal{S}$ to $\mathcal{L}$ such that the energy consumption of sensors is minimized. Assuming $|\mathcal{S}| \geq |\mathcal{L}|$, we construct a weighted complete bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$, where the weight of each edge (s_i, l_j), $s_i \in \mathcal{S}$, $l_j \in \mathcal{L}$, is calculated by $-(e_m \times d(s_i, l_j))$, where e_m is the energy cost to move a sensor in one step and $d(s_i, l_j)$ is the shortest distance between s_i and l_j. Then, we find a matching $\mathcal{M}$ in $\mathcal{G}$ with the maximum edge weights, which can be solved by the Hungarian method [16]. For each edge $(s_i, l_j) \in \mathcal{M}$, we move sensor s_i to location l_j through the shortest path (refer to [15] for details about finding the shortest path).

Reference [17] focuses on deployment with multilevel coverage. It considers two related problems: *k-coverage sensor placement* and *sensor dispatch*. The former asks how to use the minimum number of sensors in a ROI to guarantee k-level coverage. The latter asks how to dispatch mobile sensors to the designated locations computed by the placement result such that their moving energy is minimized.



(a) $r_c \leq \frac{\sqrt{3}}{2}r_s$ case (b) $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$ case

Fig. 4. The interpolating placement method.

To solve the k-coverage placement problem, [17] proposes an *interpolating placement method* based on the placement in Fig. 3(c) ($r_c < \sqrt{3}r_s$ case). Specifically, we see that a large amount of regions in each row are more than 1-covered. So, we can reuse these regions and place the least number of sensors to cover those insufficiently covered regions. Three cases are considered:

Case of $r_c \leq \frac{\sqrt{3}}{2}r_s$: In Fig. 3(c), we see that the insufficiently covered regions (marked by gray) are located between adjacent rows. If we place a new N_i row above each original O_i row by a distance of r_s, as Fig. 4(a) shows, the ROI becomes 3-covered. Here, sensors in each N_i row are separated by a distance of r_c. For $k > 3$, we can apply $\lfloor \frac{k}{3} \rfloor$ times of this 3-coverage placement and apply ($k \bmod 3$) times of the 1-coverage placement.

Case of $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$: We can add one extra N_i' row between each N_i and O_i rows to construct a 3-coverage placement, as Fig. 4(b) shows. These N_i' rows are shifted by a distance of $\frac{r_c}{2}$ and sensors are separated by a distance of $2r_c$. For $k > 3$, we can apply $\lfloor \frac{k}{3} \rfloor$ times of this 3-coverage placement and apply ($k \bmod 3$) times of the 1-coverage placement.

Case of $r_c > \frac{2+\sqrt{3}}{3}r_s$: We can duplicate k sensors on each location in Fig. 3(c).

Given a set of mobile sensors $\mathcal{S}$ and a set of locations $\mathcal{L} = \{(l_1, n_1), (l_2, n_2), \cdots, (l_m, n_m)\}$ computed by the placement result, where each location l_j will be placed with n_j sensors, [17] proposes a distributed method to dispatch $\mathcal{S}$ to $\mathcal{L}$ as follows:

1. Each sensor s_i maintains a $OCC_i[1..m]$ table, where each $OCC_i[j] = \{ (s_{j_1}, d_{j_1}), (s_{j_2}, d_{j_2}), \ldots, (s_{j_\alpha}, d_{j_\alpha})\}$, $\alpha \leq n_j$, contains the set of sensors s_{j_β}

that select l_j as their destinations and their distances d_{j_β} to l_j. Initially, $OCC_i[j] = \emptyset, \forall j$. Then, s_i selects the nearest location l_j such that $|OCC_i[j]| < n_j$ as its destination, adds $(s_i, d(s_i, l_j))$ in $OCC_i[j]$, and moves to l_j.

2. Sensor s_i periodically updates and exchanges its table with one-hop neighbors. When s_i hears the OCC_k table from a neighbor s_k, s_i combines OCC_i with OCC_k as follows: For each j, we calculate a union $U_j = OCC_i[j] \cup OCC_k[j]$. If $|U_j| > n_j$, we remove the records in U_j that have longer moving distances, until $|U_j| = n_j$. Then, we replace $OCC_i[j]$ by U_j. If s_i was in the original $OCC_i[j]$ entry, but is not in the new $OCC_i[j]$ entry, it means that s_i is replaced by other sensors with a shorter distance to l_j. Thus, s_i should reselect another destination.

3. After s_i reaches l_j, it still exchanges its table with neighbors. Since the sink will eventually observe that all locations are covered, it can notify all sensors to exit from the dispatch method.

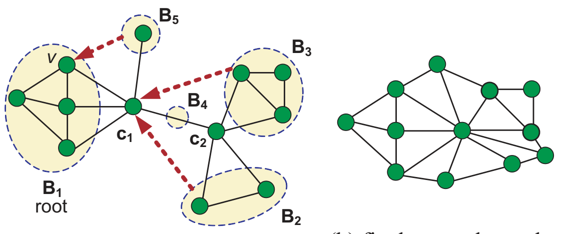
2.2. Solutions to Enhancing Coverage and Connectivity of a WSN

After deploying a WSN, some sensors may be broken or may exhaust their energy. These failed sensors may disconnect the network or cause uncovered holes. One can move some mobile sensors to relieve this problem. We introduce two such solutions for enhancing connectivity and coverage of a WSN.

2.2.1. Connectivity Enhancement

Reference [18] considers a static WSN with several isolated groups, called *islands*. To help these islands communicate with each other, we can add some mobile sensors between them. For two islands I_G and I_H, the minimum number of mobile sensors required to connect them is $M_{G,H} = \left\lceil \frac{d_{G,H}}{r_c} - 1 \right\rceil$, where $d_{G,H} = \min_{s_i \in I_G, s_j \in I_H} \{d_{i,j}\}$ is the shortest distance between I_G and I_H. Let $N(I_G)$ be the number of sensors in island I_G and $W(I_G, m)$ be the optimal set of islands that can be connected by m mobile sensors starting from island I_G. It can be derived that $W(I_G, m) = \max\{W(I_G \cup I_H, m - M_{G,H}) + N(I_G \cup I_H)\}$, where I_H is an island to be directly connected by I_G and $N(I_G \cup I_H) = N(I_G) + N(I_H) + M_{G,H}$. However, for an island I_G, if the remaining m mobile sensors cannot connect to any other island, we set $W(I_G, m) = 0$. Using dynamic programming, the minimum m to connect all islands can be found.

The work [19] considers strengthening the topology of a WSN to be *biconnected*. First, each *cut-vertex* is identified. For example, in Fig. 5(a), c_1 and c_2 are cut-vertices. By removing cut-vertices, the network is divided into several biconnected components (called *blocks*). Actually, we can 'pull' two neighboring blocks together to eliminate the cut-vertex between them. With this observation, a *block movement method* is proposed as follows: Given a network topology, we first identify all blocks along with their cut-vertices. A block can have zero, one, or multiple sensors. If two cut-vertices are directly connected, an empty block is established. Then, we can translate the network into a *block tree*, whose nodes contain blocks and cut-vertices. The block with the maximum number of sensors is the root. In Fig. 5(a), there are 5 blocks (including the empty block B_4) and 2 cut-vertices c_1 and c_2. Block B_1 is the root and blocks B_2, B_3, and B_5 are leaves. The method executes in two iterations until the network becomes biconnected: (1) Move each leaf block toward the nearest sensor of its parent block, until a new edge appears. (2) If its parent block is empty, we further move it to the upstream cut-vertex of its parent block. Fig. 5(a) gives an example, where B_5 moves toward v of its parent block B_1, and B_2 and B_3 move toward the cut-vertex c_1 since their parent block B_4 is empty. The final topology is shown in Fig. 5(b).



(a) initial network topology (b) final network topology

Fig. 5. An example of the block movement method.

2.2.2. Coverage Enhancement

The work [20] proposes a *bidding protocol* to enhance the coverage of a hybrid WSN composed of static and mobile sensors. Static sensors detect uncovered holes locally and bid for mobile sensors by the sizes of holes. It involves the following steps:

1. Each mobile sensor is assigned with a *base price*, which is an estimation of the hole size when it leaves its current position. Initially, the base price is zero for all mobile sensors. Then, mobile sensors broadcast their positions and base prices in their local areas.

2. Static sensors exchange their positions with their neighbors in two hops to construct a Voronoi diagram. If a static sensor s_i detects an uncovered hole in its Voronoi polygon, it calculates a *bid* as $\pi \times (d - r_s)^2$, where d is the distance between s_i and its farthest polygon vertex and r_s is the sensing distance. Here, the bid is an estimation of the uncovered hole size. Then, s_i sends its bid to the nearest mobile sensor whose base price is lower than the bid.

3. On receiving bids, a mobile sensor selects the highest bid and moves to cover that hole. Then, it replaces its base price by the selected bid.

The bidding protocol repeats the above steps until no static sensor can give a bid higher than the base price of any mobile sensor.

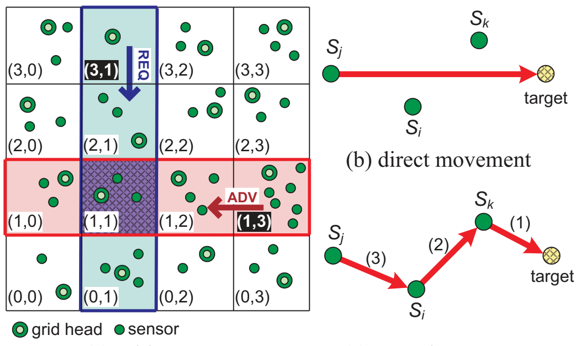
Reference [21] considers moving sensors close to locations where events could appear. Given a set of event locations, sensors are moved such that their positions can eventually approximate the event distribution. Two moving methods are proposed. In the *history-free method*, each sensor s_i at position p_i^{k-1} reacts to the appearance of an event at location l_k by moving to a new position $p_i^k = p_i^{k-1} + f_m(d(p_i^{k-1}, l_k))$, where function $f_m(\cdot)$ prohibits a sensor from passing another along the same vector in response to the same event. The *history-based method* requires sensors to maintain event history to approximate the event distribution. To maintain maximal coverage of the ROI, a sensor is not allowed to move if its movement will leave an uncovered hole.

2.3. Solutions to Assigning Mobile Sensors to Desired Locations

This may involves several issues: *sensor relocation*, *sensor navigation*, and *sensor dispatch*.

2.3.1. Sensor Relocation

Reference [22] divides a ROI into grids and moves sensors from high-density grids to low-density grids. It proposes *grid-quorum* to move sensors such that the number of exchanged messages are reduced. Specifically, a grid head is selected in each grid to maintain its information. A grid G_i with more sensors sends an *advertisement (ADV)* to its row to announce that it has extra sensors. On the other hand, a grid G_j with fewer sensors sends a *request (REQ)* message to its column to ask for extra sensors. These ADV and REQ will meet at a common grid. Fig. 6(a) shows an example, where grid $(1, 3)$ sends ADV to its row and grid $(3, 1)$ sends REQ to its column. They will meet



(a) grid-quorum (b) direct movement (c) cascade movement

Fig. 6. The grid-quorum and cascade movement methods.

at grid $(1, 1)$. This can reduce the message overhead significantly.

After identifying the targets, sensors are moved by cascaded movement rather than direct movement to prevent a single sensor from consuming too much energy. Fig. 6(b) is a direct movement, where s_j needs to travel a long distance. Fig. 6(c) is a cascaded movement, where s_k first moves to the target, then s_i moves to s_k's original position, and then s_j moves to s_i's original position.

2.3.2. Sensor Navigation

The work [23] considers navigating mobile sensors in a hybrid WSN. It assumes that all sensors do not know their own locations in the ROI. When a static sensor s_i detects an event, it will broadcast a *weight request (WREQ)* packet to search mobile sensors. On receiving WREQ, a mobile sensor m_j will bid for the event by replying its weight $w_j = \frac{A_j \times h(s_i, m_j)}{e_j}$, where A_j is the area of Voronoi polygon of m_j, $h(s_i, m_j)$ is the hop count between s_i and m_j, and e_j is the energy of m_j. A mobile sensor with a smaller weight will win the bidding.

Static sensors will then guide m_j to s_i's location. An ADV packet is sent along the path from s_i to m_j to build up a navigation field, as Fig. 7 shows. In particular, s_i sets the highest *credit* C_1 for itself. For each rebroadcast of ADV, a lower credit value will be set. Then, m_j will try to move to s_i by repeatedly searching higher credit values.

2.3.3. Sensor Dispatch

Given a set of mobile sensors $\mathcal{S}$ and a set of event locations $\mathcal{L}$, the work [24] considers dispatching $\mathcal{S}$ to $\mathcal{L}$ with a concept of load balance. Assuming $|\mathcal{S}| \geq |\mathcal{L}|$,

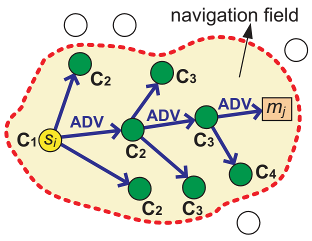


Fig. 7. Navigate a mobile sensor by credits, where $C_1 > C_2 > C_3 > C_4$.

we first calculate the energy cost $w(s_i, l_j) = e_m \times d(s_i, l_j)$ for each sensor $s_i \in \mathcal{S}$ to reach each location $l_j \in \mathcal{L}$, where e_m is the energy cost to move a sensor in one step. The scheme tries to find a matching $\mathcal{M}$ between sensors and locations by allowing a bound B_j for each $l_j \in \mathcal{L}$ as follows:

1. For each $l_j \in \mathcal{L}$, we use a bound B_j to limit the candidate sensors that l_j can match with. A sensor s_i is said as l_j's candidate if $w(s_i, l_j) \leq B_j$. Since a larger bound may lead a sensor to select a farther location, B_j will be increased gradually. Initially, each $B_j = \frac{1}{|\mathcal{L}|}\sum_{j=1}^{|\mathcal{L}|} \min_{\forall i,(s_i,l_j)\in\mathcal{S}\times\mathcal{L}}\{w(s_i, l_j)\}$.

2. For each unmatched $l_j \in \mathcal{L}$, we find a candidate sensor s_i with the minimum $w(s_i, l_j)$ to match with. If s_i is still unmatched, we add the pair (s_i, l_j) in $\mathcal{M}$. Otherwise, s_i must be matched with another location l_o. In this case, l_j will compete with l_o for s_i by three rules: (1) If $B_j > B_o$, we match s_i with l_j to avoid expanding B_j. (2) If $B_j = B_o$ and $w(s_i, l_j) < w(s_i, l_o)$, we match s_i with l_j to reduce its energy consumption. (3) If $B_j = B_o$ and s_i is the only candidate of l_j but is not that of l_o, we match s_i with l_j. When l_j wins the competition, the pair (s_i, l_o) is replaced by the new pair (s_i, l_j) in $\mathcal{M}$, and l_o becomes unmatched. Otherwise, l_j checks other candidate sensors, until there is no candidate.

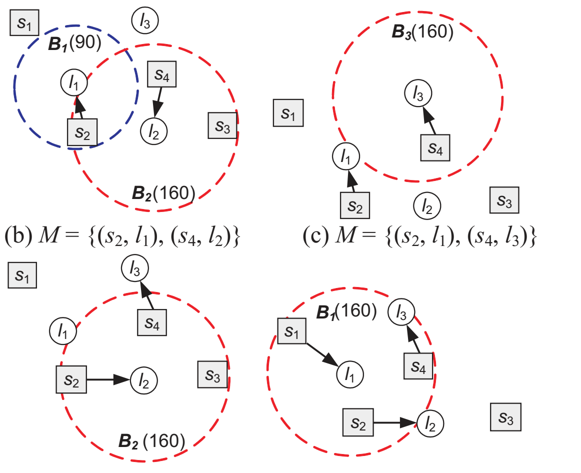
3. If l_j cannot find any match, we increase B_j by Δ_B and go to step 2, until a match is found.

4. We repeat steps 2 and 3, until each $l_j \in \mathcal{L}$ can find a sensor to match with.

Fig. 8 gives an example, where $\Delta_B = 70$. The initial bound is $\frac{79+97+94}{3} = 90$. In Fig. 8(b), l_1 matches with s_2 with bound $B_1 = 90$ and l_2 matches with s_4 with bound $B_2 = 90 + 70 = 160$. Then, after expanding B_3, l_3 finds that its candidate s_4 has been matched with l_2, so it competes with l_2 for s_4. Since $B_3 = B_2$ and $w(s_4, l_3) < w(s_4, l_2)$, (s_4, l_2) is replaced by (s_4, l_3) in Fig. 8(c). Similarly, l_2 obtains s_2 from l_1 in Fig. 8(d) and thus l_1 selects an unmatched sensor s_1. Fig. 8(e) shows the final result.

cost	s_1	s_2	s_3	s_4
l_1	105	79	238	147
l_2	219	133	153	97
l_3	181	233	177	94

(a) energy costs of mobile sensors



(b) $M = \{(s_2, l_1), (s_4, l_2)\}$ (c) $M = \{(s_2, l_1), (s_4, l_3)\}$

(d) $M = \{(s_2, l_2), (s_4, l_3)\}$ (e) $M = \{(s_1, l_1), (s_2, l_2), (s_4, l_3)\}$

Fig. 8. An example of finding the matching $\mathcal{M}$.

When $|\mathcal{S}| < |\mathcal{L}|$, a clustering approach is proposed and then the similar matching steps are executed (we omit the details).

Reference [25] considers a mobile WSN as a multi-robot system and addresses the cooperation among robots. Each robot is regarded as a resource and may be required by multiple concurrent missions. It points out that deadlock may happen when some missions never finish executing and resources are tied up, preventing other missions from starting. Then, a deadlock avoidance policy based on the *Petri nets* is proposed.

2.4. Summary of Mobility Management

Table I summarizes the mobility management methods for mobile sensors. While most methods consider a purely mobile WSN, [18, 20, 23, 24, 25] consider a hybrid WSN. References [18, 20] use mobile sensors to improve the topology of a static WSN, and [23, 24] use static sensors to detect events and send mobile sensors to event locations. For sensors' detection, [10, 12, 15, 17] consider the probabilistic sensing model. For coverage and connectivity, most deployment methods [10, 11, 13, 15, 17] address both issues, but the work [12] addresses only the coverage issue. References [20, 21, 22] move sensors to improve a WSN's coverage, while references [18, 19] move sensors to improve the network connectivity. For

references	category	hybrid WSN	probabilistic sensing	coverage issue	connectivity issue	energy issue
virtual force [10]	deployment		$\surd$	$\surd$	$\surd$	
repulsive force [11]	deployment			$\surd$	$\surd$	
gradient [12]	deployment		$\surd$	$\surd$		
Voronoi [13]	deployment			$\surd$	$\surd$	
1-coverage [15]	deployment		$\surd$	$\surd$	$\surd$	$\surd$
k-coverage [17]	deployment		$\surd$	$\surd$	$\surd$	$\surd$
island [18]	enhancement	$\surd$			$\surd$	
block tree [19]	enhancement				$\surd$	
bidding [20]	enhancement	$\surd$		$\surd$		
event motion [21]	enhancement			$\surd$		
relocation [22]	dispatch			$\surd$		$\surd$
navigation [23]	dispatch	$\surd$				$\surd$
load balance [24]	dispatch	$\surd$				$\surd$
deadlock [25]	dispatch	$\surd$				

Table I. Comparison of mobility management methods for mobile sensors.

energy concern, the dispatch solutions in [15, 17, 23] try to minimize the energy consumption of mobile sensors. Balance of energy consumption is addressed in [22, 24].

3. Path Planning of Data Ferries

3.1. Solutions to Relaying Messages by Data Ferries

Data ferries are a type of mobile sensors that are mainly designed for carrying data. For example, they can travel between isolated sensors to relay information. So, path planning is a critical issue for data ferries to minimize message delay and meet bandwidth requirements. We will discuss two types of path planning: The *adaptive planning* considers that all sensors are isolated and the final path is adjusted from the initial TSP solution. The *probabilistic planning* considers that sensors may arbitrarily roam around a ROI and ferries may meet them by a probability model.

3.1.1. Adaptive Planning

Given a data ferry $\mathcal{F}$ and a set of isolated sensors $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$, the work [26] considers how to schedule a path for $\mathcal{F}$ to visit $\mathcal{S}$. The goal is to exchange data between sensors such that the average message delay is minimized and the bandwidth requirement of each sensor is satisfied. To solve this problem, a 4-step algorithm is proposed:

1. We first calculate an initial path p by any TSP solution. The message delay between two sensors s_i and s_j on p is defined by $T^p_{ij} = \frac{|p|}{2v} + \frac{d^p_{ij}}{v}$, where $|p|$ is p's total length, v is $\mathcal{F}$'s speed, and d^p_{ij} is the distance between s_i and s_j on p. It is assumed that $\mathcal{F}$ will repeatedly travel along p. Here, $\frac{|p|}{2v}$ is the average waiting time for s_i be visited by $\mathcal{F}$, and $\frac{d^p_{ij}}{v}$ is the time for $\mathcal{F}$ to deliver s_i's data to s_j. So, the average delay incurred by p is $T^p = \frac{\sum_{1\leq i,j\leq n} b_{ij} T^p_{ij}}{\sum_{1\leq i,j\leq n} b_{ij}}$, where b_{ij} is the average amount of data to be sent from s_i to s_j.

2. Next, we try to improve p by two operations:

- *Edge Replacement:* Let $\overline{s_i s_j}$ and $\overline{s_l s_m}$ be two edges in p. Let p' be the path modified from p by replacing $\overline{s_i s_j}$ and $\overline{s_l s_m}$ with $\overline{s_i s_l}$ and $\overline{s_j s_m}$. If $T^{p'} < T^p$, then we replace p by p'.
- *Sequence Reordering:* We construct a new path p' by moving any s_i in p from its original position to another position. If $T^{p'} < T^p$, then we replace p by p'.

The above operations are repeated until no better path can be found.

3. Since the communication ranges of sensors may overlap, a *time allocation policy* Φ_p is needed to assign $\mathcal{F}$'s communication time with sensors. Specifically, we cut p into m segments $\{\xi_1, \xi_2, \cdots, \xi_m\}$ as $\mathcal{F}$ enters or leaves a sensor's communication range, and define $\Phi_p(s_i, \xi_j)$ as the portion of $\mathcal{F}$'s communication time with s_i when $\mathcal{F}$ moves along segment ξ_j. Fig. 9(a) gives an example, where the subpath from u to v is cut

into 4 segments ξ_1, ξ_2, ξ_3, and ξ_4. $\Phi_p(\cdot,\xi_1)=0$ since $\mathcal{F}$ cannot communicate with any sensor. $\Phi_p(s_2,\xi_2)=\Phi_p(s_1,\xi_4)=1$ since $\mathcal{F}$ can only communicate with one sensor. $\Phi_p(s_2,\xi_3)=\Phi_p(s_1,\xi_3)=\frac{1}{2}$ since $\mathcal{F}$ should share its time to s_1 and s_2.

4. To meet the bandwidth requirement of each s_i, $\mathcal{F}$ should spend sufficient time to communicate with s_i. If there is no sufficient time, the segments for s_i should be extended properly. For example, in Fig. 9(a), segment ξ_2 may be extended to the dotted curve to increase the communication time with s_2. Let x_j be the extra communication time of $\mathcal{F}$ to extend ξ_j and t_i be the original communication time of $\mathcal{F}$ for s_i. A linear programming is formulated to minimize the total extra communication time of $\mathcal{F}$:

$$\min \sum_{j=1}^{m} x_j,$$
$$\text{subject to } \frac{(t_i+\sum_{j=1}^{m}\Phi_p(s_i,\xi_j)x_j)\cdot R}{\frac{|p|}{v}+\sum_{j=1}^{m}x_j} \geq b_i, \quad (1)$$

where R is $\mathcal{F}$'s data rate and b_i is the bandwidth requirement of s_i. Here, the numerator and denominator are the expected amount of data that can be sent and received by s_i and the total moving time of $\mathcal{F}$ after extension, respectively. The path p after extension is $\mathcal{F}$'s traveling path.

The work [27] further considers multiple data ferries. Given n sensors and m data ferries, the goal is to find a set of paths for data ferries to visit all sensors such that the average message delay is minimized and the bandwidth requirement of each sensor is met. Four types of solutions are proposed.

Single-route algorithm (SIRA): All data ferries will move along the same path and there is no communication between them. Fig. 9(a) gives an example with two ferries. This algorithm directly extends that of [26]. For any path p, the delay to deliver data from s_i to s_j on p is $T^p_{ij}=\frac{|p|}{2mv}+\frac{d^p_{ij}}{v}$. So, the average delay of p is $T^p=\frac{\sum_{1\leq i,j\leq n}w_{ij}T^p_{ij}}{\sum_{1\leq i,j\leq n}w_{ij}}$, where w_{ij} is the weight assigned to each T^p_{ij}. Still, edge replacement and sequence reordering are applied to improve p. Finally, the linear programming in Eq. (1) can be rewritten as:

$$\min \sum_{i=1}^{n} y_i,$$
$$\text{subject to } R\cdot\frac{(2r_c+y_i)}{|p|+\sum_{j=1}^{n}y_j} \geq \frac{b_i}{m}, \quad (2)$$

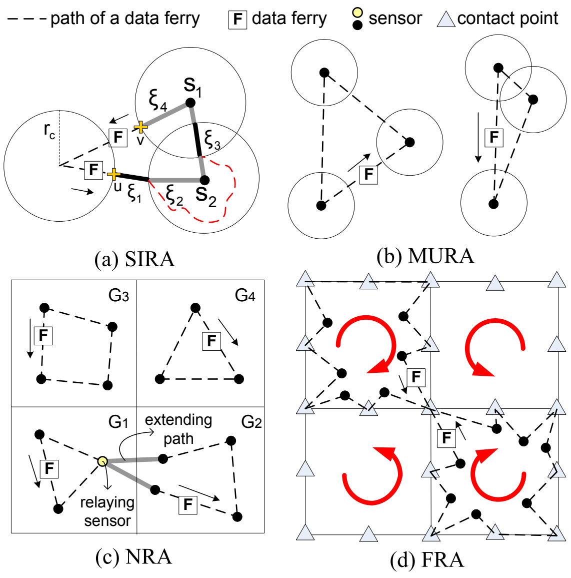


Fig. 9. Path-planning examples for data ferries.

where y_i is the extra moving length of data ferries in the communication range of s_i. In Eq. (2), the left-hand term is the product of data ferries' data rate and the ratio of data ferries' communication time allocated to s_i, and the right-hand term means that s_i's bandwidth requirement b_i is shared by m data ferries.

Multi-route algorithm (MURA): Each data ferry will move along a different path and there is no communication between them. Fig. 9(b) gives an example. In this algorithm, given a set of paths $\mathcal{P}$, we use a 2-tuple $(E_1(\mathcal{P}),E_2(\mathcal{P}))$ as the cost function to evaluate the quality of $\mathcal{P}$, where $E_1(\mathcal{P})$ is the estimated total overload of data ferries in $\mathcal{P}$ and $E_2(\mathcal{P})$ is the estimated total message delay incurred by $\mathcal{P}$ (refer to [27] for details). Intuitively, overload is the amount of data that newly appear and cannot be delivered over a time interval. Initially, we assume that each sensor has a ferry. Let $\mathcal{P}$ be the current path set and n_i be number of ferries in $p_i\in\mathcal{P}$. We adopt four operations to reduce the number of ferries and to refine the path set $\mathcal{P}$: (1) $\mathrm{overlap}(p_i,p_j)$: We extend path $p_j\in\mathcal{P}$ by including one sensor in path $p_i\in\mathcal{P}$, $p_i\neq p_j$ such that the cost is minimized. (2) $\mathrm{merge}(p_i,p_j)$: We combine p_i and p_j into one new path, and put all n_i+n_j ferries on the new path. (3) $\mathrm{merge}^-(p_i,p_j)$: This is the same as $\mathrm{merge}(p_i,p_j)$, except that we decrease the number of ferries by one. (4) $\mathrm{reduce}(p_i)$: We decrease n_i by one for p_i if $n_i>1$. We iteratively select one operation in a greedy manner to minimize the cost, until there are only m paths. After obtaining m paths, we can apply SIRA to optimize each path.

Node replying algorithm (NRA): In this scheme, each data ferry will move along a different path and static sensors will serve as relay nodes to propagate data from paths to paths. First, the ROI is divided into $c_1 \times c_2$ grids, where $c_1c_2 \leq m$, and each grid will be served by a ferry that travels on a path constructed by SIRA. Among all possible combinations of c_1 and c_2, we select the one with the minimum cost (as defined in MURA). Suppose that grids G_s and G_d want to exchange data. To relay data between them, we will try to connect G_s and G_d directly or indirectly. Two grids can be connected using the $\mathrm{overlap}(p_i, p_j)$ operation in MURA to find a relaying node. Fig. 9(c) gives an example, where there are 4 grids and 4 paths. Then, these paths will be connected by extending one to another.

Ferry relaying algorithm (FRA): Like NRA, the ROI is divided into grids, each to be served by one data ferry. Data ferries may exchange their data when they meet with each other. *Contact points* are designated along grid boundaries for this purpose, as shown in Fig. 9(d). These contact points are separated by a distance of one half of the grid boundary, so each ferry have up to eight contact points to communicate with other ferries. Data ferries of any two adjacent grids will move in reverse directions of each other. To guarantee that data ferries can meet at contact points, [27] suggests extending the path in each grid by connecting to the contact points and also extending paths such that they have the same lengths.

The work [28] considers that sensors may have different communication ranges. A data ferry only needs to *touch* any point within the communication range of each sensor to collect its data. Thus, the moving path of the ferry can be further reduced. Fig. 10 shows an example, where the ferry is initially placed at l_0. We can observe that the path $l_0 \rightarrow l_1 \rightarrow l_2 \rightarrow l_0$ is shorter than the path $l_0 \rightarrow s_1 \rightarrow s_2 \rightarrow l_0$. Here, l_1 and l_2 are called *touching points* of sensors s_1 and s_2, respectively. Based on this observation, [28] adopts evolutionary algorithms to calculate the touching points of sensors.

3.1.2. Probabilistic Planning

Given a set of mobile sensors and a data ferry $\mathcal{F}$, [29] considers planning $\mathcal{F}$'s path such that the overall probability that $\mathcal{F}$ can meet each mobile sensor is larger than a predefined threshold τ and the path length is minimized. It is assumed that these mobile sensors may move following a predefined mobility model. It is also assumed that $\mathcal{F}$ will stop at a few points for some

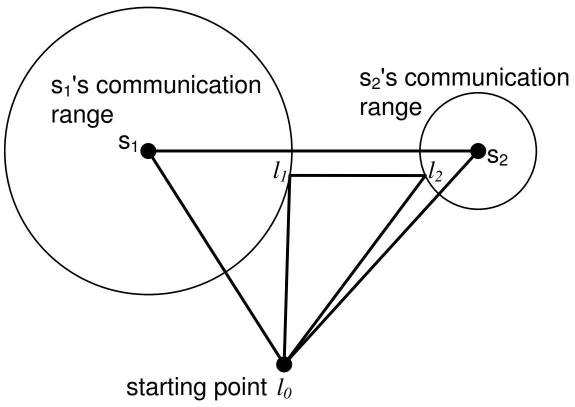


Fig. 10. An example of touching points of sensors.

periods of time when moving along the path. A 3-step solution is proposed:

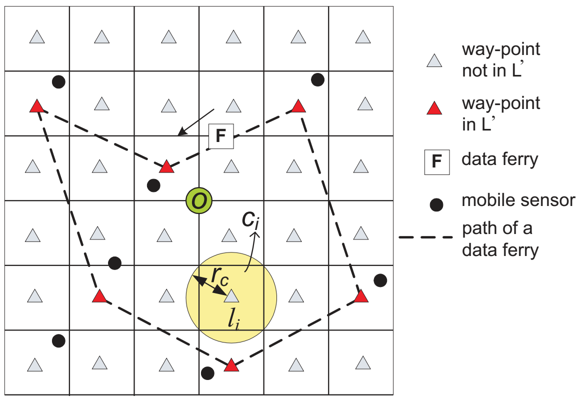


Fig. 11. An example of the probabilistic planning method.

1. We divide a ROI into grids. Let $\mathcal{L}$ be the set of the central point of each grid, called *way-point*, as shown in Fig. 11. For each $l_i \in \mathcal{L}$, let c_i be the circle centered at l_i and with a radius r_c. We define $g(c_i, s_j)$ as the *instantaneous contact probability* that $\mathcal{F}$ can meet sensor s_j inside c_i at a time instance, and $h(c_i, t_i, s_j)$ as the *time-cumulative contact probability* that $\mathcal{F}$ can meet sensor s_j inside c_i when $\mathcal{F}$ stays at l_i for a time period t_i. These probabilities depend on the mobility model of mobile sensors. In [29], these probability are developed for both a *periodic mobility model* and a *random way-point mobility model*.

2. We then select a subset $\mathcal{L}' \subseteq \mathcal{L}$ and determine the time t_i for $\mathcal{F}$ to stay in each way-point $l_i \in \mathcal{L}'$ with the following objective:

$$\min\{\sum_{l_i \in \mathcal{L}'} t_i + \beta d(l_i, O)\}, \qquad (3)$$

subject to

$$\forall s_j, \sum_{l_i \in \mathcal{L}'} h(c_i, t_i, s_j) + \max_{l_i \in \mathcal{L}'} \{g(c_i, s_j)\} \geq \tau, \quad (4)$$

In Eq. (3), O is the ROI's center, $d(l_i, O)$ is the distance between l_i and O, and β is a constant to measure the quality of the path yet to be constructed for $\mathcal{F}$. In Eq. (4), $\max_{l_i \in \mathcal{L}'}\{g(c_i, s_j)\}$ is an estimation of the meeting probability between $\mathcal{F}$ and s_j.

3. After calculating $\mathcal{L}'$ and the staying time t_i for each $l_i \in \mathcal{L}'$, we then adopt any TSP solution to construct a path to visit all way-points in $\mathcal{L}'$.

3.2. Solutions to Prolonging a WSN's Lifetime by Data Ferries

The previous section mainly focuses on using data ferries to relay data between isolated sensors. Nevertheless, with richer energy, data ferries can also help prolong the lifetime of a connected WSN. It is widely known that sensors nearby the sink could exhaust their energy faster. By scheduling data ferries to collect data from sensors, the energy consumption of sensors can be balanced and thus the network lifetime can be prolonged. We will introduce four path-planning solutions for data ferries in a connected WSN: The *recursive planning* uses a divide-and-conquer scheme to plan a ferry's path. The *tree-based planning* uses a tree structure to plan ferries' paths. The *single-hop collection* allows a ferry to directly contact each sensor. While the above solutions are centralized, the *distributed navigation* considers guiding data ferries by sensors in a distributed manner.

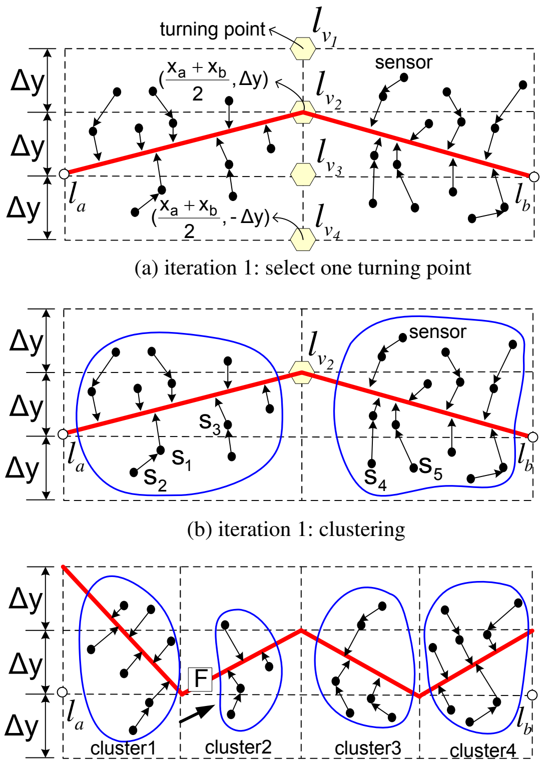
3.2.1. Recursive Planning

Given a set of sensors $\mathcal{S}$ and a data ferry $\mathcal{F}$, [30] considers planning $\mathcal{F}$'s path to visit some sensors in $\mathcal{S}$ such that the $\mathcal{F}$'s moving distance (or time) can be bounded by a pre-defined threshold, and the network lifetime is maximized. Suppose that $\mathcal{F}$ will move from a location $l_a = (x_a, y_a)$ to another location $l_b = (x_b, y_b)$. The idea is to recursively pick a turning point between l_a and l_b, until we can find a path $l_a \to l_1 \to \cdots \to l_m \to l_b$ such that the distance (or time) bounded can be meet, and the network lifetime is maximized when $\mathcal{F}$ moves along the path, where $l_1, l_2, \cdots, l_m$ are the turning points. A divide-and-conquer scheme is proposed as follows:

1. Given two locations l_a and l_b, we select a set of possible turning points such that each point locates at $(\frac{x_a+x_b}{2}, k\Delta y)$, where k is an integer and Δy is a constant such that every turning point will be inside the ROI. Among these turning points, we select the point l_v and construct a path $l_a \to l_v \to l_b$ such that the network lifetime can be maximized when $\mathcal{F}$ moves along that path (refer to [30] for the details about calculating the network lifetime). Fig. 12(a) gives an example, where there are 4 turning point and a path $l_a \to l_{v_2} \to l_b$ is constructed.

2. We divide sensors into two groups according to their distances to the line segments $l_a \to l_v$ and $l_v \to l_b$ (a sensor will favor the closer line segment). For example, in Fig. 12(b), sensors s_1, s_2, and s_3 are in one group, while s_4 and s_5 are in another group.

3. For each cluster of sensors, we recursively execute the above two steps, until the distance (or time) bounded is reached. Fig. 12(c) shows the final result, where there are two iterations.



(a) iteration 1: select one turning point

(b) iteration 1: clustering

(c) result of iteration 2

Fig. 12. An example of the recursive planning method for a data ferry.

3.2.2. Tree-Based Planning

Given a data ferry $\mathcal{F}$ and a routing tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ rooted a sink, where $\mathcal{V}$ contains all sensors $\mathcal{S}$ and the sink B, and $\mathcal{E}$ contains all tree edges, [31] considers scheduling a cyclic path for $\mathcal{F}$ to visit a subset of nodes $\mathcal{V}' \subseteq \mathcal{V}$, such that $B \in \mathcal{V}'$, the path length is not longer than $L_{\max}$, and the overall hop count along $\mathcal{T}$ from each sensor to a node in $\mathcal{V}'$ is minimized. We denote by $\delta_{TSP}(\mathcal{V}')$ the length of a path calculated by any TSP solution to traverse all nodes in $\mathcal{V}'$. This algorithm involves five following steps:

1. Initially, $\mathcal{V}' = \{B\}$.

2. Then, we construct a candidate set $\mathcal{W}$ as follows: For each $v \in \mathcal{V} - \mathcal{V}'$, we add v to $\mathcal{W}$ if $\delta_{TSP}(\mathcal{V}' \cup \{v\}) \leq L_{\max}$. If $\mathcal{W} = \emptyset$, the algorithm is terminated.

3. For each $v \in \mathcal{W}$, we calculate its *utility* by

$$U(v) = \frac{\sum_{s_i \in \mathcal{S}} d_{\mathcal{T}}(s_i, \mathcal{V}') - \sum_{s_i \in \mathcal{S}} d_{\mathcal{T}}(s_i, \mathcal{V}' \cup \{v\})}{\delta_{TSP}(\mathcal{V}' \cup \{v\}) - \delta_{TSP}(\mathcal{V}')},$$

where $d_{\mathcal{T}}(s_i, \mathcal{V}')$ is the hop count along $\mathcal{T}$ from s_i to a node in $\mathcal{V}'$. Here, the utility of v is the ratio of the reduction of total hop count that data has to be relayed along $\mathcal{T}$ to the increase of $\mathcal{F}$'s length after adding v. We then add the node with the maximum utility to $\mathcal{V}'$.

4. After adding a new node, we recalculate the utility of each $s_i \in \mathcal{V}'$. If any $s_i \in \mathcal{V}'$ has $U(s_i) = 0$, we remove it from $\mathcal{V}'$.

5. If all sensors are included in $\mathcal{V}'$, the algorithm is terminated. Otherwise, we go to step 2.

Fig. 13 gives an example. We will include s_1 and s_2 into $\mathcal{V}'$ in the first two iterations. In the third iteration, s_3 is added. Since $U(s_1)$ becomes zero, we remove s_1 from $\mathcal{V}'$. The final path is $B \rightarrow s_2 \rightarrow s_3 \rightarrow B$.

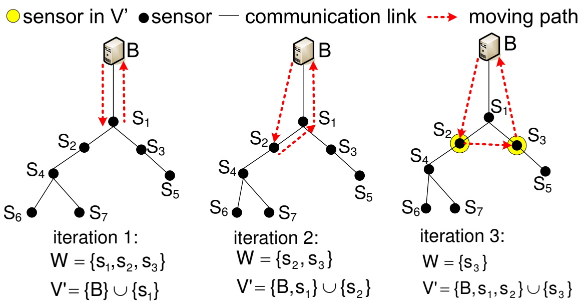


Fig. 13. An example of the tree-based planning method for a data ferry.

Reference [32] considers that sensors may aggregate their data before sending to a data ferry. It discusses how to schedule a ferry's path such that the path length is not longer than L_{max} and the total amount of data sent to the ferry can be minimized. The work suggests planning a ferry's path by adopting a *steiner tree* [33] to aggregate sensing data.

3.2.3. Single-Hop Collection

The work [34] considers planning a ferry's path to travel in a WSN such that each sensor can directly communicate with the ferry. Let $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ be a set of *candidate polling points* which contains all sensor locations and some predefined locations. Let $\mathcal{N}(l_i)$ be the set of sensors that the ferry can directly communicate with when it arrives at point $l_i \in \mathcal{L}$. The goal is to find a subset of polling points $\mathcal{L}' \subseteq \mathcal{L}$ such that each sensor belongs to at least one $\mathcal{N}(l_i)$, $l_i \in \mathcal{L}'$, and the total length is minimized. Initially, we set $\mathcal{L}' = \{B\}$, where B is the base station. Then, a greedy solution is proposed to iteratively add a polling point l_i in $\mathcal{L}$ with the minimum *covering cost* $\tau(i)$, until all sensors belongs to at least one $\mathcal{N}(l_i)$, $l_i \in \mathcal{L}'$. Here, we define $\tau(i) = \frac{\min\{d(l_i, l_j) | l_j \in \mathcal{L}'\}}{|\mathcal{N}(l_i) \cap \mathcal{U}|}$, where $\mathcal{U}$ is the set of sensors that are not in any $\mathcal{N}(l_i)$, $l_i \in \mathcal{L}'$ and $d(l_i, l_j)$ is the distance between two polling points l_i and l_j. Then, a TSP scheme can be applied for the data ferry to visit all points in $\mathcal{L}'$.

Reference [35] extends the above work by assuming that each sensor is equipped with one antenna and the data ferry is equipped with two antennas such that the dada ferry may communicate with two sensors simultaneously by a *space division multiple access (SDMA)* technology. It redefines the service coverage when the ferry stays at a location and then a similar path planning scheme in [34] is adopted.

3.2.4. Distributed Navigation

Unlike the previous centralized approaches, some efforts focus on designing a fully distributed protocol to navigate a data ferry for data collection. Given a set of sensors without location information and a data ferry with an antenna system which can accurately compute the direction of arrival (DOA) for received signals, reference [36] proposes a distributed navigation protocol to visit some representative sensors. These representative sensors are called *navigation agents (NAs)*. Then, the data ferry is navigated by the intermediate sensors between these NAs, called *intermediate navigators (INs)*. Fig. 14 shows an example. A 3-phase protocol is proposed:

1. *Identification of NAs and INs:* The set of NAs should be a dominating set of this network. The heuristic in [37] is adopted. First, a spanning tree rooted at any sensor is formed. Second, nodes mark themselves as NAs as follows:

- The root declares itself as a NA by broadcasting a *Declare-NA* message.

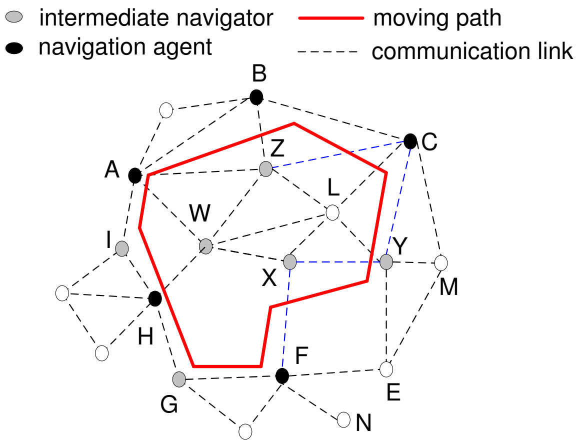


Fig. 14. An example of distributed navigation by sensors.

- When a sensor receives a *Declare-NA*, it will give up becoming a NA by broadcasting an *Accept-NA* message.
- When a sensor receives *Accept-NA* from all lower-depth neighbors, it will declare itself as a NA by broadcasting a *Declare-NA* message.

This process is repeated until each sensor is either a NA or a one-hop neighbor of a NA. Then, for each pair of NAs, the nodes passed by the shortest path (in terms of hop count) between these two NAs are marked as INs.

2. *Path computation:* A path $\mathcal{P}$ is formed to visit each NA. The work proposes adopting the *ant colony optimization-TSP* solution [38].

3. *Navigation:* Finally, the data ferry travels along $\mathcal{P}$ with the assistance of INs based on a DOA model. When visiting a NA, both NA and those sensors dominated by NA will send their data to the ferry.

Reference [39] extends the above protocol to the *k-hop data collection* scheme where sensors that are within k hops from a NA can send their data to the NA (and thus the ferry). To reduce the latency to deliver data to a NA, a sensor can pre-transmit its data to a sensor that is 1-hop away from a NA.

3.3. Summary of Path Planning

Table II summarizes the path-planning methods for data ferries. While most methods consider centralized approaches, [36, 39] uses sensors to navigate a data ferry in a distributed manner. References [30, 31, 32, 39] consider that data sent from sensors to a data ferry can be multi-hop transmission; other work [26, 27, 28, 29, 34, 35, 36] consider that ferries should directly communicate with each sensor. For the issue of communication time, [26, 27] extend the communication time of sensors to meet their bandwidth requirements, [29] minimizes the total waiting time of a ferry at each point along the path, and [35] adopts an physical layer technology to help a ferry quickly collect data from sensors. For energy concern, [30] balances the traffic loads among sensors, while [31, 32, 34, 35, 36, 39] reduce the total energy consumption of sensors. For the length concern, [30, 31, 32] give constraints on path lengths, while [26, 27, 28, 29, 34, 35, 36, 39] try to minimize path lengths.

4. Platforms and Applications of Mobile WSNs

Below, we review some interesting platforms and applications. *Mobile Emulab* [40] is a robotic testbed developed for mobile WSNs. Mobile sensors are robots that carry single-board computers and sensing devices. Remote users can control these mobile sensors in a real-time and interactive way, or through a script. Fig. 15 shows its system architecture. The video cameras will overlook the ROI and track mobile sensors. Snapshots are periodically reported to the *vision system*. Through image processing, the positions of mobile sensors are determined. The *robot system* can send motion commands to mobile sensors, which can report their sensing data to the robot system. On the other hand, the robot system can query the current positions of mobile sensors via the robot-backend system. Remote users can send *motion requests* to control mobile sensors, or send *event requests* to obtain the ROI's status.

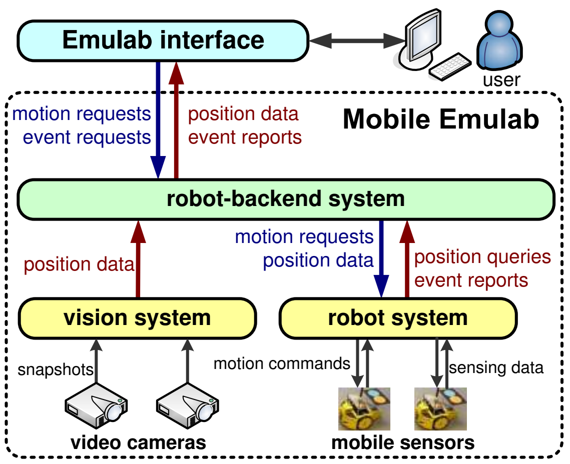


Fig. 15. The system architecture of Mobile Emulab.

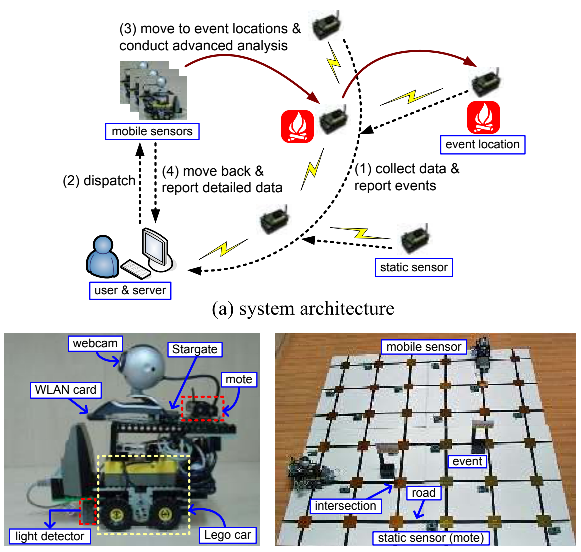
references	category	distributed scheme	multi-hop collection	communication time	energy issue	path constraint
single ferry [26]	data relaying			√		
multiple ferries [27]	data relaying			√		
touching points [28]	data relaying					
probability [29]	data relaying			√		
recursive [30]	network longevity		√		√	√
routing tree [31]	network longevity		√		√	√
aggregation [32]	network longevity		√		√	√
1-hop collection [34]	network longevity				√	
SDMA [35]	network longevity			√	√	
1-hop navigation [36]	network longevity	√			√	
k-hop navigation [39]	network longevity	√	√		√	

Table II. Comparison of path-planning methods for data ferries.

Visual surveillance systems typically collect a large amount of images from video cameras, which require a huge computation cost to analyze. Introducing the intelligence of mobile WSNs can help reduce such overheads while supporting more advanced, context-rich services. The *iMouse system* [41] is proposed to integrate static sensors and mobile sensors, as Fig. 16(a) shows. Static sensors continuously monitor the ROI and notify the server when detecting abnormal events. Once receiving such notifications, the server will dispatch mobile sensors to take snapshots at event locations. Thus, iMouse can avoid recording unnecessary images when nothing happens. Fig. 16(b) shows the components of a mobile sensor, which consists of a LEGO car carrying a MICAz mote, a webcam, an 802.11 WLAN card, and a Stargate. Fig. 16(c) shows an experimental grid-link deployment.

Robomote [42] is a mobile platform with MICA2 motes and some infrared sensors to detect obstacles. Two case studies have been tested on this platform. Based on the sensor-based path-planning scheme [43], it uses one Robomote to move along a desired contour constructed by querying neighboring sensor readings. The second case is to implement a tracking algorithm proposed in [44] to locate the light source by a Robomote.

The work in [45] uses a WSN to implement the *pursuer-evader game*. There is a moving object (called *evader*) and a data ferry (called *pursuer*). The evader roams around arbitrarily and the pursuer tries to intercept the evader based on the data reported by static sensors. One challenge for static sensors is how to quickly tell the pursuer where the evader is. To address this issue, static sensors detecting the evader



(a) system architecture

(b) platform of a mobile sensor

(c) a grid-based sensing field

Fig. 16. The iMouse system.

will elect a leader to report to the pursuer. Such reports are sent to the moving pursuer through a landmark routing [46], which operates over a tree-building mechanism. Finally, the pursuer will determine the interception path to chase the evader. The platform is developed based on MICA2 motes, an 802.11 WLAN card, and high-precision differential GPS devices.

The work in [47] proposes an implementation of data ferries. Two critical issues are addressed: (1) how to reduce the speed of a data ferry when its MAC layer encounters interference and collision, and (2) how to construct the relaying path from each sensor to a data ferry's moving path. To address the first issue, this work designs an adaptive speed control algorithm to determine whether a data ferry should slow down depending on its current data deliver rate. Specifically, a data ferry has three speeds: SLOW, STOP, and FAST. A sensor can indicate how much data that it wishes to transfer in a packet header. Then, the data ferry can select a speed accordingly. To address the second issue, a data ferry can broadcast an *interest* message to help sensors learn their distances to the data ferry's moving path.

5. Conclusions

Static WSNs have limitations on supporting multiple missions and handling different situations when network conditions change. Introducing mobility to WSNs can improve the network capability and thus relieve the above limitations. This article provides a comprehensive survey of current works on mobile WSNs. Various mobility management and path-planning schemes have been discussed. Also, several mobile platforms and applications have introduced.

Acknowledgment

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 96-2218-E-009-004, 97-3114-E-009-001, 97-2221-E-009-142-MY3, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048 and 98-EC-17-A-19-S2-0052, and by ITRI, Taiwan.

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102–114, 2002.
2. A. Rapaka and S. Madria, "Two energy efficient algorithms for tracking objects in a sensor network," *Wireless Comm. and Mobile Computing*, vol. 7, no. 6, pp. 809–819, 2007.
3. F. Hu, Y. Xiao, and Q. Hao, "Congestion-aware, loss-resilient bio-monitoring sensor networking for mobile health applications," *IEEE J. Selected Areas in Comm.*, vol. 27, no. 4, pp. 450 – 465, 2009.
4. H. Liu, P. Wan, and X. Jia, "Maximal lifetime scheduling for sensor surveillance systems with k sensors to one target," *IEEE Trans. Parallel and Distributed Systems*, vol. 17, no. 12, pp. 1526–1536, 2006.
5. M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang, "Wireless sensor networks in intelligent transportation systems," *Wireless Comm. and Mobile Computing*, vol. 9, no. 3, pp. 287–302, 2009.
6. S.S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," *Proc. IEEE Wireless Comm. and Networking Conf.*, pp. 1609–1614, 2003.
7. G. Cao, G. Kesidis, T.F.L. Porta, B. Yao, and S. Phoha, "Purposeful mobility in tactical sensor networks," *Sensor Network Operations*, 2006.
8. D. Tian and N.D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," *Proc. ACM Int'l Workshop Wireless Sensor Networks and Applications*, pp. 32–41, 2002.
9. Y.C. Wang, C.C. Hu, and Y.C. Tseng, "Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks," *Proc. IEEE Int'l Conf. Wireless Internet*, pp. 114–121, 2005.
10. Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Trans. Embedded Computing Systems*, vol. 3, no. 1, pp. 61–91, 2004.
11. N. Heo and P.K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Systems, Man, and Cybernetics–Part A*, vol. 35, no. 1, pp. 78–92, 2005.
12. N. Aitsaadi, N. Achir, K. Boussetta, and B. Gavish, "A gradient approach for differentiated wireless sensor network deployment," *Proc. IFIP Wireless Days Conf.*, 2008.
13. G. Wang, G. Cao, and T.F.L. Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.
14. F. Aurenhammer, "Voronoi diagrams–a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.
15. Y.C. Wang, C.C. Hu, and Y.C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Trans. Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.
16. H.W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.
17. Y.C. Wang and Y.C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–1294, 2008.
18. S. Zhou, M.Y. Wu, and W. Shu, "Finding optimal placements for mobile sensors: wireless sensor network topology adjustment," *Proc. IEEE Circuits and Systems Symp. Emerging Technologies: Frontiers of Mobile and Wireless Comm.*, pp. 529–532, 2004.
19. P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004.
20. G. Wang, G. Cao, P. Berman, and T.F.L. Porta, "Bidding protocols for deploying mobile sensors," *IEEE Trans. Mobile Computing*, vol. 6, no. 5, pp. 515–528, 2007.
21. Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34–42, 2003.
22. G. Wang, G. Cao, T.F.L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," *Proc. IEEE INFOCOM*, pp. 2302–2312, 2005.
23. A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 65–84, 2006.
24. Y.C. Wang, W.C. Peng, M.H. Chang, and Y.C. Tseng, "Exploring load-balance to dispatch mobile sensors in wireless sensor networks," *Proc. IEEE Int'l Conf. Computer Comm. and Networks*, pp. 669–674, 2007.
25. P. Ballal, A. Trivedi, and F. Lewis, "Deadlock avoidance policy in mobile wireless sensor networks with free choice resource routing," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 279–290, 2008.
26. W. Zhao and M.H. Ammar, "Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks," *Proc. IEEE Workshop Future Trends of Distributed Computing Systems*, pp. 308–314, 2003.
27. W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," *Proc. IEEE INFOCOM*, pp. 1407–1418, 2005.
28. B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot

routing problem in wireless sensor networks," *IEEE Trans. Knowledge and Data Engineering*, vol. 19, no. 9, pp. 1252–1261, 2007.
29. M.M.B. Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 37–48, 2006.
30. M. Ma and Y. Yang, "SenCar: an energy-efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1476–1488, 2007.
31. G. Xing, T. Wang, Z. Xie, and W. Jia, "Rendezvous planning in wireless sensor networks with mobile elements," *IEEE Trans. Mobile Computing*, vol. 7, no. 12, pp. 1430–1443, 2008.
32. G. Xing, T. Wang, W. Jia, and M. Li, "Rendezvous design algorithms for wireless sensor networks with a mobile base station," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 231–240, 2008.
33. V. Vazirani, *Approximation Algorithms*. Springer-Verlag, 2001.
34. M. Ma and Y. Yang, "Data gathering in wireless sensor networks with mobile collectors," *Proc. IEEE Int'l Parallel and Distributed Processing Symp.*, pp. 1–9, 2008.
35. M. Zhao, M. Ma, and Y. Yang, "Mobile data gathering with space-division multiple access in wireless sensor networks," *Proc. IEEE INFOCOM*, pp. 1283–1291, 2008.
36. J. Rao, T. Wu, and S. Biswas, "Network-assisted sink navigation protocols for data harvesting in sensor networks," *Proc. IEEE Wireless Comm. and Networking Conf.*, pp. 2887–2892, 2008.
37. B. Han, H. Fu, L. Lin, and W. Jia, "Efficient construction of connected dominating set in wireless ad hoc networks," *Proc. IEEE Int'l Conf. Mobile Ad Hoc and Sensor Systems*, pp. 570–572, 2004.
38. E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence*. Oxford university press, 1999.
39. J. Rao and S. Biswas, "Joint routing and navigation protocols for data harvesting in sensor networks," *Proc. IEEE Int'l Conf. Mobile Ad Hoc and Sensor Systems*, pp. 143–152, 2008.
40. D. Johnson, T. Stack, R. Fish, D.M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile Emulab: a robotic wireless and sensor network testbed," *Proc. IEEE INFOCOM*, 2006.
41. Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 60–66, 2007.
42. K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. Sukhatme, "Robomote: enabling mobility in sensor networks," *Proc. IEEE Int'l Symp. Information Processing in Sensor Networks*, pp. 404–409, 2005.
43. H. Choset, I. Konukseven, and A. Rizzi, "Sensor based planning: a control law for generating the generalized Voronoi graph," *Proc. IEEE Int'l Conf. Advanced Robotics*, pp. 333–338, 1997.
44. A. Dhariwal, G. Sukhatme, and A. Requicha, "Bacterium-inspired robots for environmental monitoring," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 1436–1443, 2004.
45. C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof, S. Sastry, and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," *Proc. IEEE European Workshop Sensor Networks*, pp. 93–107, 2005.
46. P. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 4, pp. 35–42, 1988.
47. A. Somasundara, A. Kansal, D. Jea, D. Estrin, and M. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Trans. Mobile Computing*, vol. 5, no. 8, pp. 958–973, 2006.

Authors' Biographies

You-Chiun Wang received his BEng and MEng degrees in Computer Science and Information Engineering from the National Chung-Cheng University and the National Chiao-Tung University, Taiwan, in 2001 and 2003, respectively. He obtained his Ph.D. degree in Computer Science from the National Chiao-Tung University, Taiwan, in October of 2006. Currently, he is a postdoctoral research fellow at the Department of Computer Science, National Chiao-Tung University, Taiwan. His research interests include wireless network and mobile computing, communication protocols, and wireless sensor networks. Dr. Wang served as a Local Arrangement Vice Chair in the IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS), 2007, as a Guest Editor for The Computer Journal of the special issue on "Algorithms, Protocols, and Future Applications of Wireless Sensor Networks", 2009, as a Member in the Editor Board of the IARAI International Journal on Advances in Networks and Services, 2009-present, and as TPC members of several international conferences. He is a member of the IEEE and the IEEE Communication Society.

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 obtained his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He is Professor (2000-present), Chairman (2005-present), and Associate Dean

(2007-present) at the Department of Computer Science, National Chiao-Tung University, Taiwan. He is also Adjunct Chair Professor at the Chung Yuan Christian University (2006-present).

Dr. Tseng received Outstanding Research Award, by National Science Council, ROC, twice in periods 2001-2002 and 2003-2005, Best Paper Award (Int'l Conf. on Parallel Processing, 2003), the Elite I. T. Award in 2004, and the Distinguished Alumnus Award, by the Ohio State University, in 2005. His research interests include mobile computing, wireless communication, and parallel and distributed computing.

Dr. Tseng serves on the editorial boards for *Telecommunication Systems* (2005-present), *IEEE Trans. on Vehicular Technology* (2005-2009), IEEE Trans. on Mobile Computing (2006-present), and *IEEE Trans. on Parallel and Distributed Systems* (2008-present).