Mobile Sensor Networks: System Hardware and Dispatch Software

You-Chiun Wang

Abstract—*Wireless sensor networks (WSNs)* provide a convenient way to monitor the physical environment. They consist of a large number of sensors which have sensing, computing, and communication abilities. In the past, sensors are considered as static but the network functionality would degrade when some sensors are broken. Nowadays, the emerging hardware techniques have promoted the development of mobile sensors. Introducing mobility to sensors not only improves their capability but also gives them flexibility to deal with node failure. The article studies the research progress of mobile sensor networks, which embraces both system hardware and dispatch software. For system hardware, we review two popular types of mobile sensor platforms. One is to integrate mobile robots with sensors while the other is to use existing conveyances to carry sensors. Dispatch software includes two topics. We first address how to solve different coverage problems by using a purely mobile WSN, and then investigate how to dispatch mobile sensors in a hybrid WSN to perform various missions including data collection, faulty recovery, and event analysis. A discussion about research challenges in mobile sensor networks is also presented in the article.

Index Terms—dispatch algorithms, mobility management, path planning, sensor hardware, wireless sensor and actuator network.

1 Introduction

A WSN is composed of many small, autonomous devices called *sensors*. Each sensor encapsulates sensing units, power supply (usually batteries), a microprocessor, data storage modules (such as RAM and ROM), wireless transceivers, and usually actuators [1]. It can react to surrounding stimuli like sound, light, heat, and chemicals by transforming the quantities or features of these stimuli into recordable *sensing data*. In addition to sensors, a WSN contains one or more *sinks* which take charge of collecting the sensing data in the network through multi-hop ad hoc communication. WSN substantially changes the way we monitor the physical environment. Many WSN applications have been developed, from structural health monitoring to traffic control, health care, and underground mining [2].

Traditionally, static sensors are deployed in a *region of interest (ROI)* to carry out monitoring tasks. An excessive number of sensors could be scattered over the ROI through aircraft or robots [3]. Many conventional WSN algorithms [4]–[6] then rely on *sensor redundancy* to deal with sensor failure or extend network lifetime. However, a static WSN would face many challenges [7]. First, it may not guarantee full coverage of the ROI and even network connectivity when sensors are arbitrarily scattered. Second, some subareas may be covered by only few sensors. When these sensors are broken or out of energy, the sink will no longer obtain the sensing data from the subareas. Third, some applications require multiple types of sensors and need to tactically send a certain type of sensors to particular locations [8]. This is difficult to achieve when sensors are static. Finally, some kinds of sensors are quite expensive and it is not cost-efficient to scatter too many sensors over the ROI.

Recently, thanks to the advance of MEMS (micro-electro-mechanical systems) and robotic techniques, *mobile sensors* have become possible by installing sensors on mobile platforms. *Mobile sensor networks*, which are WSNs with mobile sensors, have attracted lots of research attention. They can be classified into two categories:

- **Purely mobile WSNs:** All sensors have mobility. Usually, each sensor is *identical* in terms of sensing, computing, communication, and moving abilities. The network topology can be adaptively adjusted by moving sensors to improve monitoring quality [9] or strengthen connectivity [10].
- **Hybrid WSNs:** Few mobile sensors are added to a static WSN to improve its capability. Static sensors form the backbone for coverage and connectivity. Mobile sensors are powerful and can move to certain locations to conduct missions such as analyzing suspicious events [11] or replacing broken nodes [12].

Mobile ad hoc networks (MANETs), on the other hand, are wireless ad hoc networks where each node has mobility. At first blush, a mobile sensor network seems to be one special case of a MANET where nodes have the sensing ability. However, they are different in essence, as shown in Table 1. Both of them support multi-hop routing and network self-organization. However, there are multiple pairs of transmitters and receivers in a MANET, whereas data traffics in a mobile sensor network usually converge on the sink, which results in a *many-to-one* communication model. In addition, since sensors are small, they have limited energy and are prone to error (due to out of energy). To save their energy, in-network data processing schemes such as data aggregation or compression [13], [14] are usually adopted to reduce the amount of data sent by sensors. A mobile sensor network requires many sensors to cover an ROI, so it is difficult to assign a unique IP address to every sensor. Finally, *random waypoint* [15] is a popular mobility model in MANETs, whereas node mobility in mobile sensor networks is usually *intentional*. In other words, we can control the movement of sensors to accomplish certain missions.

Y.-C. Wang is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan. E-mail: ycwang@cse.nsysu.edu.tw

TABLE 1: Comparison on the characteristics of MANETs and mobile sensor networks.

characteristic	MANET	mobile sensor network
multi-hop routing capability	yes	yes
network self-organization	yes	yes
traffic flow	between each pair of nodes	usually converge on the sink
small size of a node	no	yes
stringent energy constraint	no	yes
node robustness	high	usually low
in-network data processing	no	yes
node density	low	high
globally unique IP addresses	yes	no
node mobility	random	usually intentional

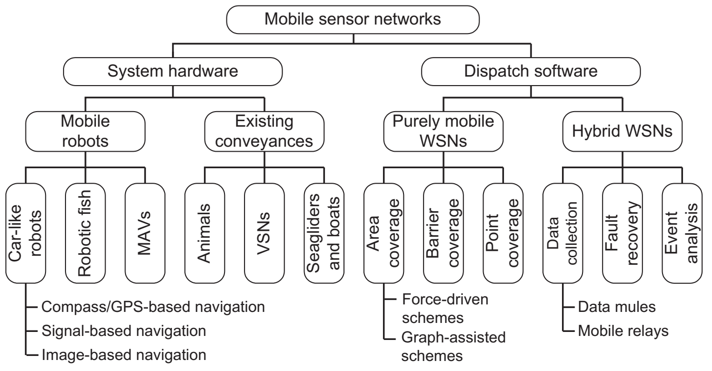


Fig. 1: Taxonomy of the research efforts surveyed in the article.

This article presents a study of research progress in mobile sensor networks from the viewpoints of both *system hardware* and *dispatch software*. Fig. 1 gives the taxonomy of our surveyed research efforts. Specifically, we first discuss the hardware development of mobile sensors, which has two contemporary trends:

- Mobile robots (Section 2): Sensors are *embedded* in mobile robots and become one critical part since they help the robots realize the surrounding conditions. For example, sensors can let a mobile robot detect a nearby obstacle so that the robot can avoid colliding with that obstacle. Depending on their locomotion models, we introduce three kinds of mobile robots. *Car-like robots*, as their name would suggest, move on a given plane to collect data. They are navigated by different means such as *compass*, *global positioning system (GPS)*, *signal*, and *image*. *Robotic fish*, on the other hand, can swim to explore aqueous environments. Finally, *micro-aerial vehicles (MAVs)* can fly and hover over an ROI for investigation purpose.
- Existing conveyances (Section 3): There are many conveyances in our life such as animals, cars, bikes, seagliders, and boats. They can move autonomously or be driven/ridden by humans without the help of sensors. However, we can use these conveyances to *carry* sensors so as to collect environmental data when they move in a desired ROI. In this way, we can save the cost to build mobile robots. When using cars or bikes as the conveyance, they may form a *vehicular ad hoc network (VANET)*. However, most VANET research aims at communication or security issues of cars [16], [17], whereas this article focuses on combining VANET and WSN where each car has the sensing capability. This combination of VANET and WSN is called *vehicular sensor network (VSN)* and can be viewed as a special case of mobile sensor network.

On the other hand, according to the categories of mobile sensor networks mentioned earlier, we address two topics in the dispatch software:

- Coverage solutions in purely mobile WSNs (Section 4): When all sensors have mobility, they can move to generate a better network topology. Since the WSN's detection capability highly depends on its coverage degree [18], a better topology means that the WSN can satisfy certain coverage requirements. Based on these requirements, we introduce three types of coverage solutions using purely mobile WSNs. *Area coverage* adopts sensors to cover every point in an ROI. This coverage requirement is basic and many schemes have been proposed, which can be classified into *force-driven* and *graph-assisted* schemes. *Barrier coverage*, on the other hand, uses sensors to check if somebody intrudes in the ROI. Finally, *point coverage*, as its name would suggest, aims at using sensors to monitor a set of points.
- Dispatch algorithms in hybrid WSNs (Section 5): Static sensors execute basic jobs such as collecting data and reporting event occurrence. Mobile sensors then tactically move to some locations to carry out various missions. Based on the missions, we survey three kinds of dispatch algorithms. First, to save energy of static sensors or maintain network connectivity, mobile sensors act as *data mules* or *mobile relays* to collect data from static sensors. Second, a WSN may be partitioned or have coverage holes due to node failure. Thus, mobile sensors conduct *faulty recovery* by restoring network coverage or connectivity. Third, mobile sensors move to event locations reported by static sensors to give *in-depth analysis of events*.

We will also make a comparison of surveyed work and discuss their challenges in each section. Table 2 summarizes the common acronyms and notations in the article.

In the literature, several studies also survey the algorithms and protocols in mobile sensor networks. [8] aims at *purposeful mobility* in WSNs, where the sensor movement is controllable so as to achieve certain missions, and it introduces the mobility-assisted sensing and routing algorithms using mobile sensors. [19] focuses on mobile sinks and relays. It discusses the motion-control techniques for mobile sinks/relays to collect data in WSNs and multihop routing protocols for sensors to reach mobile sinks/relays. [20] also surveys the data collection schemes in WSNs with mobile sinks/relays, which contains four topics: how to discover mobile sinks/relays, how to transfer data between a mobile sink/relay and a sensor, how to route data from sensors to mobile sinks/relays, and how to control the motion of mobile sinks/relays. [21] first describes

TABLE 2: Common acronyms and notations used in the article.

acronym	full name	notation	definition
GPS	global positioning system	r_s	the sensing distance of a sensor
IMU	inertial measurement unit	r_c	the communication distance of a sensor
MAV	micro-aerial vehicle	e^x	exponential function, where $e \approx 2.718$
RF	radio frequency	$d()$	the shortest distance between locations
ROI	region of interest	$gcd()$	the greatest common divisor of all parameters
TSP	traveling salesman problem	$Pr()$	probability function
VSN	vehicular sensor network	$\overrightarrow{F}$	force vector
WSN	wireless sensor network		

the challenges caused by sensor or sink mobility at the link layer and then surveys mobility-aware MAC protocols to address these challenges. Obviously, these studies merely address the software aspect of mobile sensor networks by describing related algorithms and protocols. Compared with them, this article wants to give a *full view* of the research progress in mobile sensor networks, which is carried out by discussing hardware development of mobile sensor platforms and software (algorithm) design to dispatch mobile sensors for various missions. Our previous work [7] targets at both mobile sensors and relays. It first discusses how to assign mobile sensors to deploy a WSN, to improve coverage and connectivity, and to visit certain sites. Then, it surveys path planning protocols for mobile relays to collect messages and extend network lifetime. Finally, it investigates few applications using mobile sensors. The differences between this article and [7] are threefold. First, this article systematically introduces the development of mobile sensor platforms which includes not only mobile robots but also existing conveyances. Second, comparing with [7], our survey of dispatch software covers the extra topics of barrier coverage, point coverage, and faulty recovery. Third, we present detailed comparisons between surveyed work and address possible research challenges. This part is ignored in [7].

2 Hardware: Mobile Robots

Numerous studies develop their own mobile robots and embed sensors in these robots to detect the environment. Mobile robots are usually small but can perform 2D or 3D movement to explore a given ROI. According to the taxonomy in Fig. 1, below we discuss three types of mobile robots: car-like robots, robotic fish, and MAVs.

2.1 Car-like Robots

Car-like robots equipped with wheels for locomotion are popular mobile platforms for sensors. They can move on a 2D plane and detect nearby obstacles to avoid collision. However, it is critical to navigate the robots, so below we introduce three common navigation techniques. In *compass/GPS-based navigation*, a robot knows its heading direction or position by its digital compass or GPS receiver, respectively. In *signal-based navigation*, robots estimate their positions via *radio frequency (RF)* and ultrasonic signals. Finally, in *image-based navigation*, robots are guided by some image patterns.

2.1.1 Compass/GPS-based Navigation

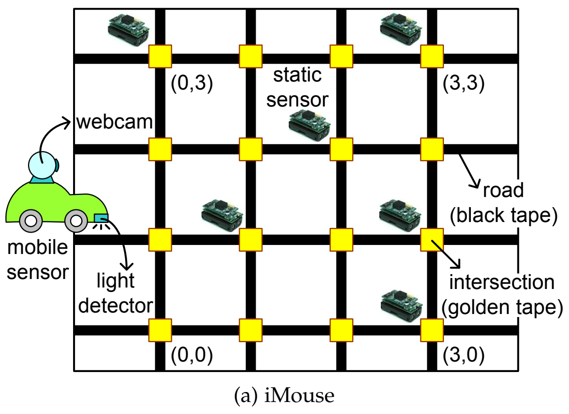
Robomote [22] is a mobile platform designed to carry a *mote* device. It has an Atmel microcontroller, two motors, a digital compass, and infrared transceivers. The mote is the master of Robomote, which controls the motion of each motor (and the corresponding wheel) via the Atmel microcontroller. Users can develop TinyOS applications to manage the behavior of Robomote. The digital compass is used for heading and must be calibrated periodically. This can be done by asking Robomote to make a *full turn* to detect the maximum and minimum readings of the compass and set them as reference. Robomote also emits infrared rays on either end to detect obstacles. When the received infrared signal exceeds a threshold, the mote is aware of obstacles and makes a detour to avoid collision.

COMET [23] is a hardware testbed developed to implement and evaluate cooperative control techniques. It consists of ten mobile robots, each based on a Tamiya radio-controlled trunk. The trunk has a size of 498 mm × 375 mm × 240 mm and supports a load of 3.7 kg. Each robot is equipped with a GPS receiver and IMU (inertial measurement unit) sensors for navigation and an infrared module for obstacle avoidance. It also uses a camera and a laser range finder to create a 2D map of the distance to surrounding objects. COMET has some practical applications:

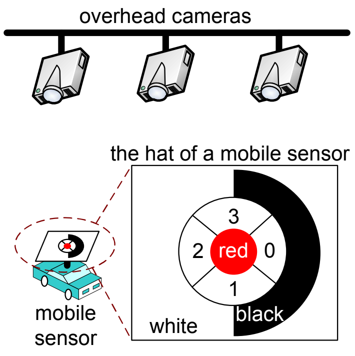
1) Formation control: Robots move along a trajectory and keep the desired formation shape. Each robot should follow a distant leader without colliding with neighbors.
2) Flocking: Robots interact with each other to reach a consistent state in their heading angles and inter-robot separation distances. Each robot should align its heading based on the average of its heading and the heading of its neighbors.
3) Perimeter detection and tracking: Robots are asked to find and track a dynamic perimeter such as a building, chemical substance spill, or landmark.
4) Target assessment: Each robot first secures the path to the target, then drives to the target, and finally encircles the target.

2.1.2 Signal-based Navigation

The work of [24] develops a miniature mobile sensor platform for condition monitoring of structures such as civil infrastructure, transportation systems, and industrial plants. Mobile sensors move inside the structure and conduct three inspections to check structural health: 1) *eddy-current inspection* searches service-induced fatigue and stress-corrosion cracks, 2) *magnetic-flux-leakage inspection* finds material loss due to corrosion, gouging, or pitting, and 3) *ultrasonic inspection* checks the interior volume of structure. To localize mobile sensors, a beacon-listener system is built. Each mobile sensor is equipped with a Cricket transmitter [25], which broadcasts beacons with its identifier on an RF channel and an ultrasonic pulse simultaneously. Cricket receivers are mounted inside the structure to hear the beacons. Each receiver uses the *time difference of arrival (TDOA)* between an RF signal and an ultrasonic pulse sent by the same mobile sensor to compute its distance to that mobile sensor. When three or more receivers know their distances to



(a) iMouse



(b) AMRST

Fig. 2: Image-based navigation for mobile sensors.

a mobile sensor, they can use *trilateration* to localize the mobile sensor.

The study of [26] adopts mobile sensors to *collaboratively* track targets plying in an ROI. Each mobile sensor keeps a table to record the target information and periodically exchanges the table with neighbors. If a mobile sensor has no target to track, the mobile sensor remains stationary until it detects a target or learns new targets by exchanging the table. When a mobile sensor finds that its target is tracked by too many mobile sensors, it switchs to track another target. Infrared modules are equipped on mobile sensors to avoid colliding with each other. To localize mobile sensors, static beacon nodes (with known coordinates) are installed on the ceiling. These beacon nodes continually send their identifiers via ultrasonic pulses and RF signals. When a mobile sensor hears the RF signals and ultrasonic pulses from at least three beacon nodes, it can use TDOA and trilateration to calculate its position.

2.1.3 *Image-based Navigation*

iMouse [27] supports *event-driven* indoor surveillance applications by using static sensors to monitor an ROI and mobile sensors to provide event analysis. For example, static pressure sensors are deployed in a room. Once some static sensors report something intruding into the room, mobile sensors equipped with cameras can move to the event locations to take snapshots for further analysis. A prototype of mobile sensors is developed, which includes a Lego car to support mobility, a mote to talk to static sensors, a webcam to take snapshots, a WiFi interface to send images to the remote sink, and a small embedded computer (called Stargate) to control the behavior of the mobile sensor. The navigation of mobile sensors is realized by sticking different colors of tape on the ground, as shown in Fig. 2(a). Black tape represents roads while golden tape indicates intersections. Each mobile sensor has a light detector in its front to project light on the ground and receive the reflection. Every intersection has a unique coordinate and thus a mobile sensor knows its position by detecting golden tape.

AMRST [28] uses overhead cameras to monitor the positions and orientations of mobile sensors, as shown in Fig. 2(b). Each mobile sensor has a unique color pattern placed on top of it (called *hat*). Since the body of a mobile sensor is hidden below its hat, the cameras can search for the appropriate pattern to find the mobile sensor. Specifically, each hat has a central red circle and two rings. The red circle is to identify the sensor's *existence*. The outer ring containing a black semi-circle is to decide the sensor's *orientation*. The inner ring, which is divided into four quarter-circles marked as digits 0 to 3 in Fig. 2(b), is to obtain the sensor's *identification*. Each quarter-circle is drawn by one of the three colors: white, blue, and green, which represent numbers 0, 1, and 2, respectively. Thus, AMRST supports $3^4 = 81$ mobile sensors, each with a unique hat pattern.

2.2 Robotic Fish

Mobile underwater WSN is an emerging networking paradigm to monitor water bodies such as lakes or seas. Several research efforts have developed prototypes of *robotic fish* to implement mobile underwater WSNs. However, RF signals cannot travel over long distances in an underwater environment [29]. In this case, conventional positioning techniques such as GPS or RF-based trilateration may not be applied to localize robotic fish. Instead, the robotic fish can adopt other schemes such as vision processing or acoustic propagation to obtain positioning information.

The work of [30] adopts two robotic fish to track one target (a water polo) in a swimming tank. Each fish has one tail fin for forward/backward swimming and two pectoral fins for turning. It also has a camera (with visual angle of 120°) installed at the mouth position for recognition purpose. To track the target and avoid colliding with the other fish, four situations are considered to define the fish's behavior, as shown in Fig. 3:

1) NFT (no fish and target) situation: Fish A does not see the target and fish B. In this case, fish A searches for the target by turning clockwise or counterclockwise based on the last position of the target in its field of view.
2) OF (only fish) situation: Fish A sees only fish B. Thus, fish A turns to the opposite side of fish B to search for the target.
3) OT (only target) situation: Fish A sees only the target. In this case, fish A adjusts its orientation toward the direction of the target and swims to the target.
4) BFT (both fish and target) situation: Fish A sees both the target and fish B. Thus, fish A swims to the target while avoids colliding with fish B by combining the attractive force from the target and the repulsive force from fish B.

The study of [31] develops a robotic fish with one tail fin propelled by an *ionic polymer-metal composite* actuator. It is a type of electro-active polymers that generate large bending movement with only several volts [32]. Thus, the fish can swim for a long time without recharging. Each fish has an RF antenna and a pair of buzzer and microphone for ranging measurement

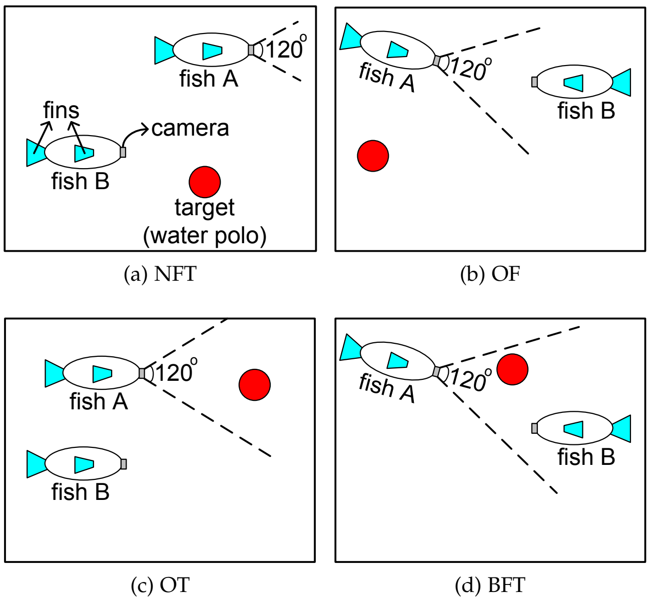


Fig. 3: Four situations encountered by fish A.

between it and a fixed beacon node or another fish. This allows localization with respect to known locations or relative localization between two fish. Ranging is realized by concurrent use of an RF signal and an acoustic pulse. RF signals propagate (in air) at 3×10^8 m/s while acoustic pulses propagate (in water) at 1.5×10^3 m/s. By measuring time difference between the reception of RF signal and acoustic pulse, a fish can compute its distance to the beacon source. However, since RF signals propagate poorly underwater, the fish conducts ranging only when it surfaces with its RF antenna exposed in air (but the buzzer and microphone are still underwater). When the fish swims into deep water, extra devices such as accelerometer, gyroscope, and pressure sensor (to get depth data) can help localize it.

2.3 MAVs

MAVs are emerging as a novel class of mobile sensor networks capable of navigating semi-automatically in unknown environments. Each MAV can fly in 3D space but carry less weight due to its small size. Flying with a suspended load is a challenging task since the load changes the MAV's flight characteristics. Thus, [33] adaptively alters the *center of gravity (CoG)* of an MAV with four rotors to reduce the swing of its load during flight. Let $\{\mathcal{B}\}$ be the moving aircraft-fixed coordinate system. The coordinate of the MAV's CoG is denoted by a vector $\mathbf{r} = [x_G\, y_G\, z_G]^T$, which represents the distance from the origin of $\{\mathcal{B}\}$ to the MAV's CoG. When the MAV is balanced, we have $\mathbf{r} = 0$. However, a suspended load produces extra forces and torques acting on the MAV, resulting in a shift ρ in $\mathbf{r}$. By using dynamic programming, the MAV can compute the proper height and position of each rotor to eliminate the effect of ρ so that the vector $\mathbf{r}$ is always kept zero during flight.

Simbeeotic [34] is a Java simulator used to model the behavior of an MAV swarm that collaboratively sense to explore an ROI. Each MAV should deal with obstacle avoidance, navigation, path planning, and environmental manipulation. In Simbeeotic, the *virtual* environment and MAV bodies are composed from simple shapes (e.g., box, cone, and sphere) and complex geometries (e.g., triangular mesh and convex hull). The kinematic state of each MAV is simulated by integrating the effects of gravity, rotor thrust, and wind applied to that MAV. Physical interactions between objects, such as environmental manipulation by a robot or bump sensors, are modeled by a *3D continuous collision detection module*. Besides, Simbeeotic uses a small-scale helicopter testbed for real-world experiments. The wall-mounted cameras track the position and orientation of each helicopter and send the information to Simbeeotic. Then, the kinematic state of its MAV in the simulation is updated accordingly. By sending commands via RF transceivers, Simbeeotic can also control the behavior of helicopters such as adjusting their flying directions.

The work of [35] considers coordinating an MAV swarm to cover an ROI. The goal is to reduce the amount of sensing overlap between MAVs. Each MAV is equipped with a 3-axis accelerometer and a 3-axis gyro to obtain orientation, an ultrasonic ranger to estimate altitude, an optical flow sensor to measure velocity, and an IEEE 802.15.4 radio to receive RF signals. The MAV swarm consists of two types of nodes: *anchors* and *explorers*. Anchors are MAVs that land in the ROI (using a dispersion algorithm to determine their landing positions at initialization). Explorers are MAVs that fly to sense the environment. They collect RF signatures from nearby anchors and send the signatures to a remote sink, which computes the ROI's probabilistic coverage. Then, the sink directs the flight of explorers to improve the overall coverage.

2.4 Discussion on Mobile Robots

Table 3 compares the mobile robots surveyed in Section 2. Car-like robots are popular and most studies develop small robots powered by small-capacity batteries. Thus, they cannot travel in a long distance and are demonstrated in small testbed areas. For instance, Robomote [22] is powered by two 1.5V AAA batteries, so it lasts for just 25 minutes in motion. Only [23] develops a large robot powered by a large-capacity lithium-ion battery. The robot can move in several hundred meters and work well outdoors. However, it encounters a higher hardware cost. Broadly speaking, *scalability* is a big challenge for car-like robots. In a purely mobile WSN (as we will discuss in Section 4), all nodes have mobility and we require hundreds to thousands nodes to form the WSN. Thus, the *hardware cost* limits the scalability of robots. On the other hand, although we require fewer robots in a hybrid WSN (as we will mention in Section 5), they may have to traverse the whole WSN deployed in a huge ROI many times. In this case, both *moving speed* and *energy* limit the scalability of robots, since we expect robots to quickly visit many locations to conduct their missions.

Robotic fish swim underwater and thus it brings more challenges. [30] aims at target tracking and collision avoidance between fish. This is realized by vision processing, but the technique may not work well in deep water. [31] designs a special tail fin to save the energy spent on swimming, and it also uses static beacon nodes to localize the fish. However, since the propagation delay of a signal is much longer underwater than in the air, many negative effects such as path loss, multipath problem, and interference become more significant. Thus, how to eliminate these effects and avoid collision of synchronizing signals so as to provide accurate localization of fish deserves further investigation.

MAVs allow sensors to fly in 3D space and they can fast move to target locations as compared with car-like robots. Collision avoidance is especially critical for them during flight

TABLE 3: Comparison on the features of different mobile robots.

research effort	navigation technique	robot length/weight	testbed area	collision avoidance	hybrid WSN
Car-like robots:					
work of [22]	compass	N/A	$1.22\,\text{m} \times 2.44\,\text{m}$	✓	✓
work of [23]	GPS	$\approx 50\,\text{cm}$	outdoor	✓	
work of [24]	signal	N/A	$2\,\text{m} \times 1\,\text{m}$	✓	✓
work of [26]	signal	$\approx 15\text{cm}$	$3\,\text{m} \times 3\,\text{m}$	✓	✓
work of [27]	image	$\approx 24\,\text{cm}$	$1.5\,\text{m} \times 1.5\,\text{m}$		✓
work of [28]	image	N/A	$3.5\,\text{m} \times 4\,\text{m}$		
Robotic fish:					
work of [30]	image	N/A	$2.25\,\text{m} \times 1.25\,\text{m}$	✓	
work of [31]	signal/compass	$\approx 21\,\text{cm}$	$22.5\,\text{m} \times 13\,\text{m}$		✓
MAVs:					
work of [33]	image	N/A	$3\,\text{m} \times 3\,\text{m}$	✓	
work of [34]	image	$\approx 20\,\text{cm}$	$7\,\text{m} \times 6\,\text{m}$	✓	
work of [35]	signal	$< 30\,\text{g}$	$5\,\text{m} \times 3\,\text{m}$	✓	✓

(to prevent them from crashing). Unlike other two studies, [35] lets some MAVs land in the ROI to serve as static sensors to navigate other (flying) MAVs by using RF signals. MAVs also face many challenges. For example, how to control them without human interaction is a challenging task, because flight is much difficult than moving on a 2D plane. Besides, most MAVs can carry loads of less than several hundred grams. This means that MAVs have severe energy limitation due to their small battery weight. Thus, it is an open research issue to save the energy of MAVs so as to extend their flight time.

3 Hardware: Existing Conveyances

Instead of building mobile robots, some studies use existing conveyances to carry sensors. They can move autonomously or be controlled by people. Following the taxonomy in Fig. 1, we introduce three types of conveyances. First, animals are equipped with sensors for management or tracking purposes. Then, cars or bikes (which form a VSN) move along streets and bring sensors to monitor urban areas. Finally, seagliders and boats carry sensors to inspect the ocean by swimming into or sailing the water.

3.1 Animals

Sensors can be put on animals to monitor wildlife or manage livestock. Specifically, light-weighted, battery-powered tracking devices, called *collars*, are attached to animals' necks to record their migration or limit their movement. A collar has small memory chips to store data and may be equipped with actuators to give stimuli to the animal for the control purpose.

ZebraNet [36] helps biologists to track wild zebras at the Mpala Research Center in Kenya. Each collar (attached on a zebra) has a GPS device to localize its zebra and an RF transceiver to send data to a remote sink in a multihop manner. To provide the biologists with an accurate view into the daily migration pattern of zebras, the GPS device should take readings every eight minutes to obtain sufficient samples. The RF transceiver operates at 900 MHz and has a communication range of up to five miles. To save the energy, radio communication occurs every two hours to reduce the radio's duty cycle. ZebraNet adopts a *flooding protocol* to transmit data. Specifically, every two hours each collar activates its RF transceiver and searches for other collars in its communication range. Then, it sends as many buffered data as possible to neighbors. After five minutes, the collar turns off its RF transceiver to save energy. Any unsent data will wait for the next interval for transmission.

Virtual fence [37] uses an insulated wire (called *fencing wire*) to let domestic animals stay in a bound area. This is realized by using a generator unit to supply the fencing wire with a low-intensity current to create an *electromagnetic field (EMF)* around the wire. Animals are attached with electronic collars to detect the EMF. Once they approach the fencing wire, their collars inform the animals via some stimuli. Based on the EMF's intensity, three zones are defined (from inner to outer):

1) Standby zone: Collars remain idle to save their energy.
2) Warning zone: When an animal moves into this zone, the collar emits an audio signal to prevent it from moving forward.
3) Exclusion zone: This zone is closest to the fencing wire. When an animal enters the exclusion zone, the collar applies an electric stimulus to force it to move back.

Each collar is powered by four AAA batteries, so electric stimuli never hurt the animal. Experiments are conducted on 145 animals including dairy, beef cattle, and horses for 800 days. They show that after one or two electric stimuli, animals can know the range of exclusion zone, which demonstrates the feasibility of virtual fence.

Wildsensing [38] is an animal-monitoring project to analyze the social co-location patterns of European badgers residing in a woodland habitat. Unlike ZebraNet, since GPS signals may be weak in woodland, wildsensing uses RFID (radio frequency identification) to track badgers. It contains three components:

1) Collars and detection nodes: Each badger wears a collar with a 433 MHz active RFID tag. Detection nodes, which are composed of RFID receivers and sensors, are deployed in *hotspots* (e.g., badger setts and latrines) to detect badgers. RFID's detection range is at most 30 m.
2) Static sensors: They are deployed in woodland to monitor temperature and humidity. Static sensors and detection nodes form a WSN via IEEE 802.15.4 links.
3) Mobile sink: It is solar powered and collects data from the WSN. The sink can transmit data to end users via 3G links.

A sensor has only 1M-byte memory, so wildsensing uses a *delta-based compression method* to store data. It first takes an RFID/sensor reading as the base. Then, the difference between the base and each subsequent reading is stored to save the memory.

3.2 VSNs

VSN is the combination of VANET and WSN by equipping each vehicle (e.g., car or bike) with sensors. VANET research mainly aims at packet routing among vehicles or security/privacy concern of vehicles. On the contrary, VSN research focuses on using vehicles to carry sensors so as to monitor a large geographic area with fewer sensors. This can be carried out by car/bike mobility to cover different regions in different times.

CarTel [39] is a VSN developed to visualize the data collected from sensors located on cars. Specifically, each car gathers and processes street information locally and then delivers it to a remote sink, where the information is stored in a central database for further analysis and visualization. In CarTel, each car is equipped with a GPS receiver, a camera, and a WiFi interface. The GPS receiver allows each car to measure delays along road segments in order to infer traffic congestion. The cameras capture street view to build applications which help navigate drivers in unfamiliar environments. Cars can exchange their information via WiFi interfaces. When a car meets a road-side WiFi AP (access point), it can send the recorded information to the sink through a backbone network. On the other hand, the sink can send queries or commands to certain cars via APs and the VSN formed by cars.

BikeNet [40] uses sensors installed on a bike to gather quantitative data related to the cyclist's ride. It not only gives context to the cyclist performance (e.g., riding speed, distance traveled, and calories burned) but also collects environmental conditions for the ride (e.g., the degrees of pollution, allergen, noise, and terrain roughness of a given route). Each bike is equipped with the following sensors: microphone, magnetometer, pedal speed monitor, inclinometer, lateral tilt, stress monitor (to measure galvanic skin response), speedometer, CO_2 meter, and GPS receiver. They share an IEEE 802.15.4 channel to organize a VSN. Besides, the cyclist carries mobile phones that have cameras to take snapshots and GSM/WiFi interfaces for communication. When a cyclist rides near a GSM/WiFi base station, the mobile phones send sensor readings to the sink for analysis. The analyzed data can be used in communal projects such as pollution monitoring and cyclist experience mapping.

MobEyes [41] is an urban-monitoring platform built on VSN by equipping cars with cameras and chemical sensors. It considers collecting information from cars about criminals that spread toxic chemicals in some parts of a city. Each car periodically generates a *summary chunk* which includes its position (5 bytes), timestamp (2 bytes), recognized license plates (by the camera, each with 6 bytes), and sensing data with 10 bytes (e.g., the concentration of potential toxic agents). Every 65 chunks are packed into one single 1500-byte *summary* to be disseminated to neighboring cars. This allows a squad car to *opportunistically* harvest summaries from encountering cars and therefore generates a *distributed meta-data index* for forensic purposes (e.g., crime scene reconstruction or crime tracking).

The work of [42] adopts cars to monitor CO_2 concentration in urban areas. Each car is equipped with a CO_2 sensor, a GPS receiver, and a GSM module. Cars periodically report their monitoring CO_2 concentration to the sink via GSM short messages. However, CO_2 concentration could vary over different regions and time, and it incurs extra charges to send GSM short messages. Thus, to reduce the communication cost, the ROI is divided into *grids* and the sink adjusts the reporting rates of the cars in each grid. Specifically, when the variation of CO_2 concentration is higher or there are fewer cars in a grid, the sink should give a faster reporting rate to that grid. In this case, it can get more samples to reflect the drastic change of CO_2 concentration. On the other hand, when the variation is smaller or there are more cars in a grid, the sink can give a lower reporting rate to that grid in order to reduce the communication cost.

Installing sensors on cars also helps develop vehicular safety applications. For instance, [43] installs a GPS receiver and IMU sensors on a car to measure its *driving data* (e.g., heading direction, position, velocity, acceleration, and yaw rate). Cars use an *unscented Kalman filter* [44] to process driving data and exchange filtered data by sending beacons in every 250 ms. Besides, *event messages* are immediately broadcasted once an urgent event is detected (e.g., a car is braking). Thus, each car can predict the paths of its neighbors by taking as input a state vector which comprises filtered values of driving data and the road curvature information from a digital map's database. This can improve road safety by avoiding collision between cars or letting a driver flash the brake lights in case of an emergency.

3.3 Seagliders and Boats

Some studies use seagliders to carry sensors to monitor the ocean. They can dive to the deep ocean to explore unknown environments. For example, [45] develops a 1.5 m (in length) by 21.3 cm (in diameter), 52 kg, torpedo-shaped seaglider. It is driven by a variable buoyancy system and automatically operates in coastal and open-ocean scenarios by adjusting the volume-to-weight ratio. The seaglider has two side wings to control the direction. It swims at an average speed of 0.4 m/s and can reach a depth of 200 m. It also has an RF interface and Iridium satellite modem for communication. Subsurface navigation is achieved by adopting a compass, altimeter, and internal dead reckoning. Besides, the seaglider is equipped with CTD (conductivity, temperature, and depth) sensors, a 3-channel fluorometer, and a 3-channel backscattering detector. Thus, it can monitor the oceanic health by analyzing chlorophyll, colored dissolved organic matter, and rhodamine.

Smart sensor web [46] is an ocean-observing platform that uses two subsystems to gather the oceanic information: *mooring sensor system* (i.e., static sensors) and *seagliders* (i.e., mobile sensors). The mooring sensor system serves as the infrastructure for communication and provides precise timing throughout the water column. Each node is anchored on the seabed (with a depth of 900 m) and has a near-surface float at a depth of 165 m with a suite of sensors. It monitors float/mooring dynamics and water stresses via acoustic devices and gyro-enhanced orientation sensors. On the other hand, a seaglider can dive to 1000 m and move horizontally at around 0.5 knot using 0.5 W of power. It has a GPS receiver, an Iridium satellite antenna, and an altimeter for navigation. Since RF waves attenuate rapidly underwater, each seaglider has a hydrophone to communicate with others via acoustic signals.

The work of [47] provides high-accuracy positioning of a boat under the problem of poor GPS signal reception due to the weather or noise. It proposes a *sensor integration solution* that takes the boat's dynamics into account. Specifically, the boat carries acoustic transducers and a laser range finder to survey nearby obstacles (e.g., breakwaters). IMU sensors can obtain the boat's pendular data, which consists of a triaxial accelerometer and three single-axis rate gyros mounted along three orthogonal axes. Besides, the boat is equipped with one magnetometer to record the environmental data such as gravity and earth magnetic field. The above data are fed into an extended Kalman filter to correct the inaccuracy of the GPS readings. This improves position and attitude estimation for the boat.

3.4 Discussion on Existing Conveyances

Table 4 compares the conveyances discussed in Section 3, where the selection of conveyance depends on the application. The benefit to use animals as conveyances is that they can be easily tracked or managed by sensors. Since animals may live in various habitats, different localization techniques are adopted. Specifically, [36], [37], and [38] use GPS, EMF, and RFID to localize the animals residing in grassland, rangeland, and woodland, respectively. Energy is critical in these applications since animals can only wear light-weighted collars with small batteries. [38] also uses static sensors to monitor woodland conditions, so a hybrid WSN is formed. One major challenge is that the movement of each single animal is uncontrollable. Thus, some animals (and their collars) may be *isolated* and cannot exchange data with others. This case is similar to a *delay tolerant network* [48] and it deserves further investigation to apply (and modify) the current data forwarding methods in such networks to these applications.

Compared with other conveyances, VSNs have some advantages. First, cars and bikes are the most common transport, so it is easy to construct a VSN. Second, their moving patterns are predictable or even controllable. This property is suitable for mobile sensor networks. Third, energy is usually not a concern in VSNs[1], so cars can carry sophisticated sensors and adopt complicated algorithms. However, VSNs essentially inherit the research challenges from VANETs [49] when we aim at the *communication* aspect. Besides, the mission of a VSN may be dynamically changed and each node in the VSN may not store all possible missions in advance due to its limited memory size. In this case, how to efficiently perform *reprogramming* on some or all nodes will be a challenging task [50].

Using seagliders and boats to carry sensors could help scientists to systematically explore the ocean by constructing a mobile sensor network to provide long-term monitoring of the ocean. It also makes *underwater WSNs* become practical. [45], [46] adopt the static mooring system that contains an inductive power module to charge the batteries of seagliders. On the other hand, [47] uses low-power sensors to conserve energy. However, the underwater environment significantly extends the propagation delay of an RF signal and could even absorb/refract the signal. This brings some interesting research topics: 1) *How to develop an efficient underwater communication protocol?* 2) *How to provide accurate localization of a seaglider, as the GPS signal may become very weak underwater?*

4 SOFTWARE: COVERAGE SOLUTIONS IN PURELY MOBILE WSNS

Coverage is a research issue peculiar to WSNs, which distinguishes them from wireless ad hoc networks. Each sensor has two distances r_s and r_c to determine whether it can detect an event and talk to others, respectively. Coverage problems have been extensively studied in WSNs and can be classified into three categories [51]:

1. [40] is an exception because bikes usually do not provide energy. Thus, sensors have to be powered by AAA batteries.

- *Area coverage problems:* Given an ROI within which all points are treated equally, they ask to deploy a WSN in the ROI to satisfy both *complete coverage* where every point in the ROI is monitored by k sensors and *network connectivity* where no sensors are isolated [18]. When $k = 1$, the problem is called *1-coverage problem*; otherwise, it is called *k-coverage problem*.
- *Barrier coverage problems:* Given a thin belt-area (called *barrier*), they ask to deploy sensors in the barrier so that every intruder can be detected by at least k sensors before it enters the ROI [52]. When $k = 1$, this problem is called *strong 1-barrier coverage problem*; otherwise, it is called *strong k-barrier problem*.
- *Point coverage problems:* They use sensors to cover a set of discrete space points, which can be *points of interest* to represent an ROI (e.g., event locations) or used to model physical targets. In static WSNs, the goal is use sensors to permanently monitor all points. In mobile WSNs, the goal is to let sensors regularly move to visit these points to guarantee that they are covered for certain periods.

Below, we survey the coverage solutions by using a purely mobile WSN, where all sensors have mobility and they can adjust the topology to satisfy coverage requirements.

4.1 Area Coverage Problems

Area coverage is critical since it affects the event detection capability of a WSN [53]. Many schemes use a purely mobile WSN to solve such problems, which are generally divided into two categories. *Force-driven schemes* view sensors as electric charges so that they can exert forces on each other to move. *Graph-assisted schemes* adopt graph-theory or geometric approaches to calculate where to move sensors.

4.1.1 *Force-driven Schemes*

They aim at solving the 1-coverage problem. Each sensor s_i is exerted by three kinds of *virtual forces*: $\overrightarrow{F_{iA}}$ by the ROI, $\overrightarrow{F_{iO}}$ by obstacles, and $\overrightarrow{F_{ij}}$ by a sensor s_j. A force is either *attractive* (positive) or *repulsive* (negative). Let $\mathcal{S_F}$ be the set of sensors that exert forces on s_i. Then, the *combined force* on s_i is computed by

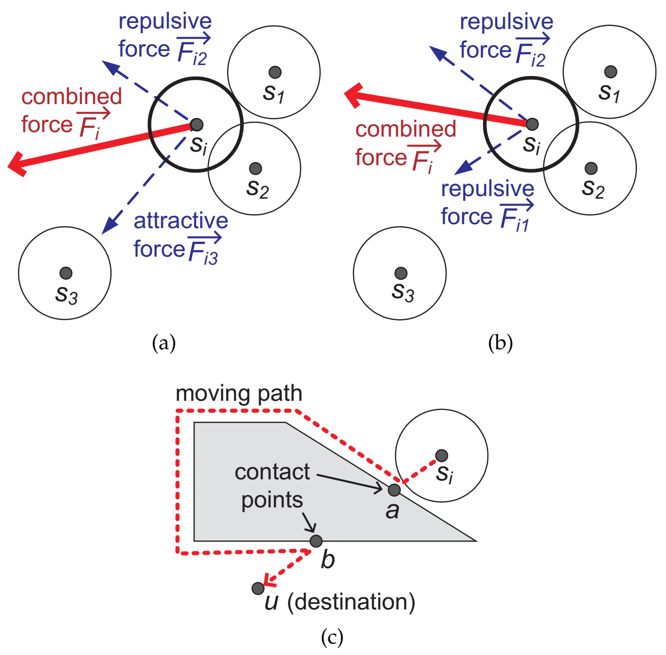
$$\overrightarrow{F_i} = \overrightarrow{F_{iA}} + \overrightarrow{F_{iO}} + \sum\nolimits_{j\in\mathcal{S_F}} \overrightarrow{F_{ij}},$$

where the orientation of $\overrightarrow{F_i}$ is decided by the vector sum of all forces exerting on s_i. Sensor s_i is *iteratively* moved by $\overrightarrow{F_i}$ until it enters two states. In the *oscillation* state, s_i moves back and forth in a small region many times. In the *stable* state, s_i moves less than a small distance in a period. Thus, s_i stops moving in both cases.

The work of [54] models an ROI by grids, so $\overrightarrow{F_{iA}}$ is decided by *preferential area grids* A_1, A_2, $\cdots$, A_m that exert attractive forces on a sensor s_i. It is computed by the distance $d(s_i, A_1, A_2, \cdots, A_m)$ between s_i and these grids. Similarly, $\overrightarrow{F_{iO}}$ is decided by *obstacles grids* O_1, O_2, $\cdots$, O_n that exert repulsive forces on s_i. It is computed by the distance $d(s_i, O_1, O_2, \cdots, O_n)$ between s_i and these grids. Every sensor exerts a force on s_i, so $\mathcal{S_F}$ contains all sensors (except s_i). Each force $\overrightarrow{F_{ij}}$ exerted by a sensor s_j on s_i is either attractive or repulsive based on the distance $d(s_i, s_j)$. The force is expressed

TABLE 4: Comparison on the features of different conveyances used to carry sensors.

research effort	application	conveyance	localization	energy issue	hybrid WSN
Animal:					
work of [36]	track zebras	zebra	GPS	✓	
work of [37]	pasturage	livestock	EMF	✓	
work of [38]	track badgers	badger	RFID	✓	✓
VSNs:					
work of [39]	road survey	car	GPS		
work of [40]	bike riding	bike	GPS	✓	
work of [41]	find chemical	car	GPS		
work of [42]	monitor CO_2	car	GPS		
work of [43]	car safety	car	GPS		
Seagliders and Boats:					
work of [45]	explore ocean	seaglider	GPS		✓
work of [46]	explore ocean	seaglider	GPS		✓
work of [47]	localize boats	boat	GPS/IMU	✓	



repulsive force $\overrightarrow{F_{i2}}$
combined force $\overrightarrow{F_i}$
attractive force $\overrightarrow{F_{i3}}$
s_1
s_2
s_3
s_i
(a)
repulsive force $\overrightarrow{F_{i2}}$
combined force $\overrightarrow{F_i}$
repulsive force $\overrightarrow{F_{i1}}$
s_1
s_2
s_3
s_i
(b)
moving path
contact points
s_i
a
b
u (destination)
(c)

Fig. 4: Examples of force-driven schemes, where the forces from the ROI and obstacles are ignored for simplification: (a) attractive and repulsive forces, (b) only repulsive forces, and (c) bypassing an obstacle using the right-hand rule.

by a polar coordinate system (R, θ), where R and θ are its magnitude and orientation:

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

where d_{th} is a threshold distance, θ_{ij} is the orientation of $\overrightarrow{s_i s_j}$, and w_A and w_R are measures of the attractive and repulsive forces, respectively. Fig. 4(a) gives an example. Since $d(s_i, s_1) = d_{\text{th}} = 2r_s$, $\overrightarrow{F_{i1}} = 0$. Then, s_2 exerts a repulsive force $\overrightarrow{F_{i2}}$ while s_3 exerts an attractive force $\overrightarrow{F_{i3}}$ on s_i due to $d(s_i, s_2) < d_{\text{th}}$ and $d(s_i, s_3) > d_{\text{th}}$, respectively.

The study of [55] tries to *uniformly* distribute sensors over an ROI without obstacles, so $\overrightarrow{F_{iA}} = \overrightarrow{F_{iO}} = 0$. Sensors are viewed as electrons and they repel with each other by the Coulomb's law in physics, where $\overrightarrow{F_{ij}} \geq \overrightarrow{F_{ik}}$ if $d(s_i, s_j) \leq d(s_i, s_k)$. In addition, $\overrightarrow{F_{ij}} = F_{\max}$ when $d(s_i, s_j) \approx 0$, and $\overrightarrow{F_{ij}} = 0$ when $d(s_i, s_j) > r_c$ (this indicates that $\mathcal{S}_{\mathcal{F}}$ contains all s_i's one-hop neighbors). When $0 < d(s_i, s_j) \leq r_c$, the force from s_j to exert on s_i is computed by

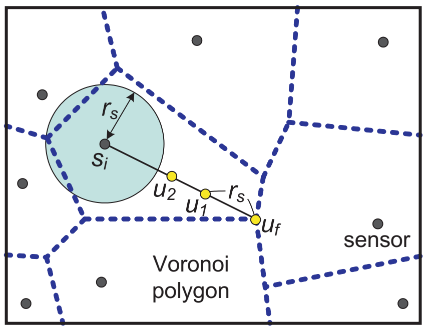
$$\overrightarrow{F_{ij}} = \frac{\mu_i}{\mu^2}(r_c|p_i - p_j|)\frac{p_j - p_i}{|p_j - p_i|},$$

where μ_i is the sensor density in s_i's communication range, μ is the expected sensor density, and p_i and p_j are the positions of s_i and s_j, respectively. Fig. 4(b) gives an example, where $d(s_i, s_3) > r_c$. Both s_1 and s_2 exert repulsive forces on s_i but s_3 does not affect s_i since it is outside s_i's communication range.

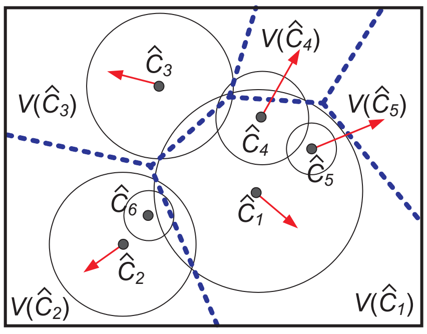
The work of [56] defines an origin $(0, 0)$ at the ROI's center to attract each sensor s_i by a force $\overrightarrow{F_{iA}} = \overrightarrow{(x_i^2 + y_i^2)^{1/2}}$, where (x_i, y_i) is s_i's position and the orientation of $\overrightarrow{F_{iA}}$ is from (x_i, y_i) to $(0, 0)$. The ROI may have obstacles but they do not exert forces on s_i, so $\overrightarrow{F_{iO}} = 0$. Instead, when s_i encounters an obstacle, it moves along the obstacle's boundary by the *right-hand rule*. Fig. 4(c) shows how to bypass an obstacle, where u is s_i's destination. Points a and b are the obstacle's contact points along $\overline{s_i u}$. When meeting a, s_i keeps contact with the obstacle, until it reaches b. Since $\mathcal{S}_{\mathcal{F}}$ contains all s_i's one-hop neighbors, s_i is exerted by a repulsive force $\overrightarrow{F_{ij}} = \overrightarrow{K(1/d^2(i, j) - 1/r_c^2)}$ from each neighbor s_j, where K depends on r_c (e.g., $K = 1000r_c^2$). Fig. 4(b) gives an example, where s_i is exerted by the repulsive forces from its neighbors s_1 and s_2.

4.1.2 *Graph-assisted Schemes*

A *Voronoi diagram* can present the proximity information related to a set of geometric nodes [57]. It is formed by perpendicular bisectors of lines that connect any two nearby nodes, as shown by dotted lines in Fig. 5(a). Each point in a *Voronoi polygon* is closer to the node in this polygon than to any other nodes. [58] uses this property to solve the 1-coverage problem by finding *coverage holes* and moving sensors to reduce them. Specifically, each sensor s_i exchanges its position with neighbors to construct a Voronoi polygon. Then, s_i checks if it can completely cover its polygon. If not, there must be a coverage hole. Thus, two methods are proposed to move s_i to reduce that hole, as shown in Fig. 5(a). In the *Voronoi-based method*, s_i moves toward the farthest vertex u_f of the polygon, and stops at the point u_1 such that $d(u_1, u_f) = r_s$. In the *minimax method*, s_i also moves toward u_f, but stops at the point u_2 whose distance to all vertices of the polygon is the minimum. Point u_2 is identified by checking all circles that pass through any two or three vertices of the polygon. Among these circles, the one with the shortest radius that covers all vertices of the polygon is selected, and its center is thus u_2.



(a) Voronoi diagram



(b) Voronoi-Laguerre diagram

Fig. 5: Move sensors by graph-theory methods.

The minimax method reduces the moving distance of sensors but incurs higher complexity, as compared with the Voronio-based method.

The work of [59] solves the 1-coverage problem by heterogeneous sensors with different sensing ranges. A modified Voronoi diagram, called *Voronoi-Laguerre diagram* [60], is used to move sensors. Given a circle $\hat{C}_i$ with center c_i and radius r_i and a point u, the *Laguerre distance* between $\hat{C}_i$ and u is defined by $d(\hat{C}_i, u) = d^2(c_i, u) - r_i^2$. When u lies inside $\hat{C}_i$, $d(\hat{C}_i, u) < 0$. Then, a *Voronoi-Laguerre polygon* $V(\hat{C}_i)$ is formed by the set $\mathcal{U}$ of all points such that $d^2(\hat{C}_i, u) \leq d^2(\hat{C}_j, u)$, $\forall u \in \mathcal{U}$ and $\forall j \neq i$. Fig. 5(b) gives an example. Some polygons, such as $V(\hat{C}_6)$, are *null* as they do not appear in the diagram. We can replace each circle $\hat{C}_i$ by a sensor s_i with sensing distance r_i to find coverage holes in the diagram. Then, the following method guides sensors to move to reduce the holes. If a sensor s_i has a null polygon, it is in an overcrowded area. Thus, s_i stops moving since it cannot improve coverage. Otherwise, s_i moves toward a point p in its polygon such that p has the minimum distance to all vertices of the polygon. Fig. 5(b) gives an example, where arrows indicate the moving directions of sensors. Sensor s_6 does not move since it has a null polygon $V(\hat{C}_6)$.

The study of [61] addresses the 1-coverage problem by solving two subproblems: *sensor placement* and *sensor dispatch*. The placement problem asks to use the fewest sensors to achieve 1-coverage of the ROI. Based on the placement solution (a set of target locations $\mathcal{L}$), the dispatch problem asks to move sensors to $\mathcal{L}$ so that their moving energy is minimized. The ROI is partitioned into subregions based on obstacles. Then, two sensor placement patterns in Fig. 6(a) and (b) solve the placement problem in each subregion. When $r_c \geq \sqrt{3}r_s$, adjacent sensors are separated by $\sqrt{3}r_s$, so both coverage and connectivity are guaranteed. When $r_c < \sqrt{3}r_s$, adjacent sensors on each row are separated by r_c to keep the row's connectivity. Since adjacent rows have a distance of $\left(r_s + \sqrt{r_s^2 - \frac{1}{4}r_c^2}\right) > r_c$, a column of sensors is required to connect every row. Then, the dispatch problem is translated into a *maximum-weight maximum-matching problem*. Given a set of sensors $\mathcal{S}$, a weighted complete bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$ is constructed. The weight of each edge $(s_i, (x_j, y_j))$, $s_i \in \mathcal{S}$, $(x_j, y_j) \in \mathcal{L}$ is computed by $w(s_i, (x_j, y_j)) = -E_m \times d(s_i, (x_j, y_j))$, where E_m is the energy cost to move a sensor in one unit distance and $d(s_i, (x_j, y_j))$ should consider obstacles. By using the Hungarian method [62], a maximum-weight maximum-matching $\mathcal{M}$ is obtained from $\mathcal{G}$. Finally, each sensor s_i is dispatched to location (x_j, y_j) if $(s_i, (x_j, y_j)) \in \mathcal{M}$.

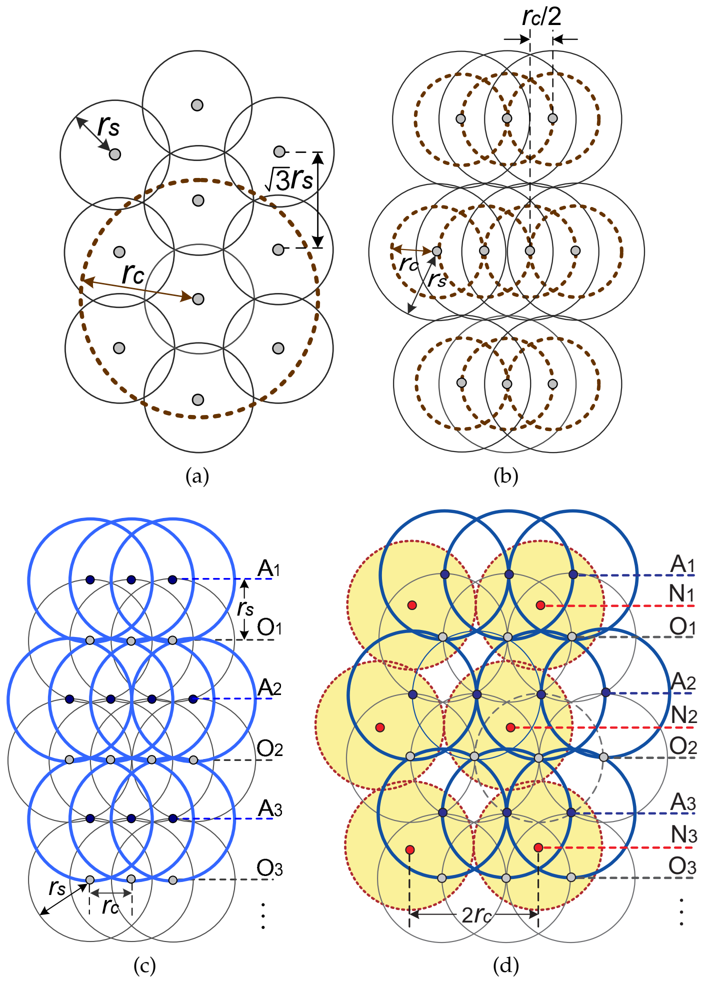


Fig. 6: Sensor placement patterns: (a) 1-coverage when $r_c \geq \sqrt{3}r_s$, (b) 1-coverage when $r_c < \sqrt{3}r_s$, (c) 3-coverage when $r_c \leq \frac{\sqrt{3}}{2}r_s$, and (d) 3-coverage when $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$.

The work of [63] aims at the k-coverage problem by addressing two subproblems: *k-coverage placement* and *distributed dispatch*. The k-coverage placement problem asks to use the fewest sensors to achieve k-coverage of the ROI. The distributed dispatch problem asks sensors to contend for the target locations so that they consume less energy. To solve the k-coverage placement problem, three cases are considered:

1) $r_c \leq \frac{\sqrt{3}}{2}r_s$ case: In Fig. 6(c), all O_i rows together provide 1-coverage. When an A_i row is placed above each O_i row by a distance of r_s, the ROI is 3-covered. If we apply $\lfloor k/3 \rfloor$ times of the 3-coverage placement and $(k \bmod 3)$ times of the 1-coverage placement, the k-coverage placement $(k > 3)$ can be obtained.

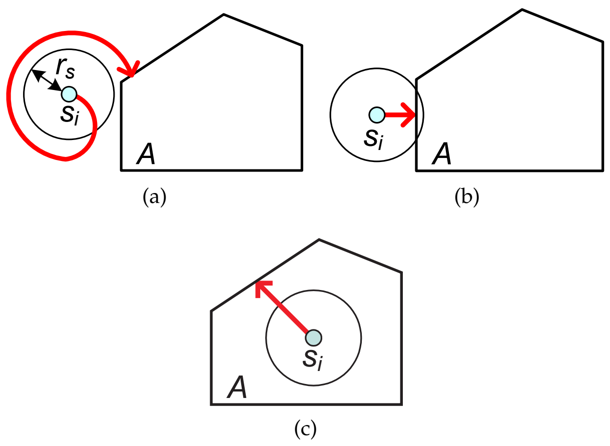


Fig. 7: Three cases to seek the boundary of an ROI $\mathcal{A}$: (a) s_i is outside the boundary, (b) s_i is on the boundary, and (c) s_i is inside the boundary.

2) $\frac{\sqrt{3}}{2}r_s < r_c \leq \frac{2+\sqrt{3}}{3}r_s$ case: We can add an N_i row between each A_i and O_i rows to make the ROI 3-covered. Fig. 6(d) gives an example, where the coverage of each N_i sensor is a circle filled with color and two adjacent N_i sensors have a distance of $2r_c$. For $k > 3$, the placement is realized by the same method in the above case.
3) $r_c > \frac{2+\sqrt{3}}{3}r_s$ case: We can apply k times of the 1-coverage placement in Fig. 6(a) and (b) to obtain the k-coverage placement.

Let $\mathcal{L} = \{(l_1, n_1), (l_2, n_2), \cdots, (l_m, n_m)\}$ be the set of target locations, where each location l_j has n_j vacancies for sensors, $j = 1..m$. To solve the distributed dispatch problem, each sensor s_i keeps an $OCC_i[1..m]$ table, where an entry $OCC_i[j] = \{(s_{j_1}, d_{j_1}), (s_{j_2}, d_{j_2}), \cdots, (s_{j_k}, d_{j_k})\}$, $k \leq n_j$, indicates every sensor s_{j_α} choosing l_j as the destination and its distance d_{j_α} to l_j. Initially, $OCC_i[1..m]$ is empty. Then, s_i chooses the closest location l_j such that $|OCC_i[j]| < n_j$ (i.e., l_j still has a vacancy) as the destination. Thus, a record $(s_i, d(s_i, l_j))$ is added to $OCC_i[j]$ and s_i moves toward l_j. Each sensor periodically exchanges its table with neighbors. When obtaining a table $OCC_k[1..m]$ from another sensor, s_i combines its $OCC_i[1..m]$ table with $OCC_k[1..m]$ by $OCC_i[j] = OCC_i[j] \cup OCC_k[j]$, $j = 1..m$. After the combination, if $|OCC_i[j]| > n_j$ (i.e., too many sensors compete for l_j), the records with longer distances to l_j are discarded until $|OCC_i[j]| = n_j$. Then, s_i checks if its record (s_i, l_j) is still in $OCC_i[j]$. If not, s_i loses the competition and has to choose another location as the new destination. When all locations in $\mathcal{L}$ have no vacancy, s_i stops moving.

4.2 Barrier Coverage Problems

Barrier coverage has applications such as border surveillance and immigrant management. Some studies adopt a purely mobile WSN to solve the barrier coverage problems. Specifically, [64] uses mobile sensors to achieve strong k-barrier coverage by forming k *sensor-disjointed chains* to surround an ROI $\mathcal{A}$. A sufficient condition to achieve the maximum k value is that n sensors are uniformly placed on the convex hull of $\mathcal{A}$. Thus, we have $k \leq (2nr_s)/L_c(\mathcal{A})$, where $L_c(\mathcal{A})$ is the shortest length of a barrier chain. Then, a two-step distributed method is developed for mobile sensors to surround $\mathcal{A}$. In step 1, each sensor s_i seeks $\mathcal{A}$'s boundary (shown in Fig. 7):

1) If s_i's sensing range does not overlap with $\mathcal{A}$, s_i is outside $\mathcal{A}$'s boundary. Thus, s_i moves along a *spiral* to seek $\mathcal{A}$ with equal probability on all directions.
2) If s_i's sensing range has partial overlap with $\mathcal{A}$, s_i is on $\mathcal{A}$'s boundary. Thus, s_i approaches the boundary via the shortest path.
3) If s_i's sensing range is included by $\mathcal{A}$, s_i is inside $\mathcal{A}$'s boundary. Thus, s_i moves in a straight line after selecting a direction to the boundary.

In step 2, sensors form barriers around $\mathcal{A}$ by *virtual forces*. When s_i has two neighbors s_j and s_k, they exert forces to move s_i such that $d(s_i, s_j) = d(s_i, s_k)$. If s_i has only one neighbor s_j, s_j exerts a force to move s_i such that $d(s_i, s_j) = 2r_s$. Thus, all sensors can evenly surround $\mathcal{A}$ and achieve strong k-barrier coverage.

Suppose that sensors move in a barrier by the *random direction mobility model* [65], where a sensor arbitrarily selects the moving direction and its speed is randomly chosen from $[0, v_{\max}]$. Given intruders that attempt to cross the barrier, [66] derives the *k-barrier coverage probability* $Pr(\Lambda \geq k)$, where Λ is the cumulative coverage count by mobile sensors. The problem is similar to the kinetic theory of gas molecules in physics [67], where mobile sensors are viewed as air molecules and intruders are viewed as electrons. A mobile sensor has to *collide* with an intruder for detection. We can use the kinetic theory to compute the average traveling distance of an intruder between successive detection by mobile sensors. Let v_i be the speed of an intruder and $\overline{v}_{\text{rel}}$ be the average relative speed of mobile sensors with respect to intruders. Given the density of mobile sensors μ and an observation period τ, the *coverage rate* is $\Theta_v = \mu \cdot \varphi \cdot \overline{v}_{\text{rel}}$, and the *average uncovered distance* is computed by the traveling distance of an intruder divided by the number of sensor coverage (which equals to the intruder speed divided by the coverage rate):

$$\lambda = \frac{v_i}{\Theta_v} = \frac{v_i}{\mu \cdot \varphi \cdot \overline{v}_{\text{rel}}}, \quad \text{where} \quad \varphi = \frac{2r_s + \pi r_s^2}{v_i \tau}.$$

The probability that an intruder meets j mobile sensors along its traveling path is

$$Pr(\Lambda = j) = \frac{e^{-\Theta_v \cdot \tau}(\Theta_v \cdot \tau)^j}{j!}.$$

Therefore, the k-barrier coverage probability can be derived by

$$\begin{aligned} Pr(\Lambda \geq k) &= 1 - \sum_{j=0}^{k-1} Pr(\Lambda = j) \\ &= 1 - \frac{\int_{\Theta_v \cdot \tau}^{\infty} t^{k-1} e^{-t} dt}{\int_0^{\infty} t^{k-1} e^{-t} dt} = 1 - \frac{\Gamma(k, \Theta_v \cdot \tau)}{\Gamma(k)}, \end{aligned}$$

where $\Gamma(k, \Theta_v \cdot \tau)$ and $\Gamma(k)$ are the incomplete Euler gamma function and the Euler gamma function, respectively. Then, by adjusting the number of mobile sensors (i.e., changing μ), the desired k-barrier coverage probability can be obtained.

The work of [68] considers that sensors are not enough to achieve strong 1-barrier coverage and thus exploits their mobility to increase the *intruder detection probability*. Given a belt area, intruders may cross it from one boundary to another. Let m be the minimum number of sensors required to provide strong 1-barrier coverage and n be the number of mobile sensors, where $n < m$. Intruders arrive stochastically at each point i, $i = 1..m$, and time is divided into *slots*. At any point, the *intruder inter-arrival time* x is a random variable in the Weibull distribution [69]:

$$\text{density function:} \quad f(x) = \frac{\beta}{\lambda}\left(\frac{x}{\lambda}\right)^{\beta-1} e^{-\left(\frac{x}{\lambda}\right)^{\beta}},$$

$$\text{cumulative function:} \quad F(x) = 1 - e^{-\left(\frac{x}{\lambda}\right)^{\beta}},$$

where $x \geq 0$, $\lambda > 0$, and $\beta \geq 1$. When $\beta = 1$, the Weibull distribution degrades to the Poisson distribution. Let a_i^t denote the state of intruder arrival, where $a_i^t = 1$ if an intruder arrives at point i in slot t and $a_i^t = 0$ otherwise. Also, let b_i^t be the state of sensor presence at point i, where $b_i^t = 1$ if a sensor stays at point i in slot t and $b_i^t = 0$ otherwise. Then, the goal is to increase the intruder detection probability

$$P_{\text{detection}} = \lim_{t' \to \infty} \frac{\sum_{t=1}^{t'} \sum_{i=1}^{m} a_i^t b_i^t}{\sum_{t=1}^{t'} \sum_{i=1}^{m} a_i^t},$$

while minimize the average moving distance L_{move} of sensors. Two schemes to let mobile sensors patrol in the belt area are proposed. In the *periodic monitoring scheduling scheme*, each sensor at point j, $j = 1..m$, moves to point $(j + n) \bmod m$ and stays at the new point for T slots. The iteration is repeated until all sensors exhaust their energy. Let $m' = m/gcd(m, n)$ and $n' = n/gcd(m, n)$. Then, this scheme guarantees that

$$P_{\text{detection}} = \frac{n}{m} \quad \text{and} \quad L_{\text{move}} = \frac{2r_s(mn' + nm' - 2nn')}{m'T}.$$

The *coordinated sensor patrolling scheme* considers that the points with higher intruder arrival probabilities should be selected first. A sensor is marked *available* if it detected an intruder in the previous slot and *unavailable* otherwise. Each unavailable sensor should stay at its current point since an intruder may arrive soon. Then, available sensors are assigned to the points without sensors but with higher intruder probabilities. Thus, L_{move} can be minimized by the scheme.

4.3 Point Coverage Problems

Some studies use a purely mobile WSN to cover a set of points in the time domain. These points can be event locations, points of interest, or physical objects in the ROI. Sensors will move to visit the points to make them covered for a portion of time.

Given a set of locations where events appear, [9] addresses how to move sensors such that their positions can approximate to the event distribution. Two schemes are thus developed. In the *history-free scheme*, a sensor s_i reacts to event k occurring at location (x_k, y_k) by updating its position $p_i^k = p_i^{k-1} + f(D_i)$, where $D_i = d((x_k, y_k), p_i^{k-1})$. Here, p_i^{k-1} is s_i's position and $f(\cdot)$ should satisfy three criteria:

1) $0 \leq f(D_i) \leq D_i$, which means that s_i should never move past event k.
2) $f(\infty) = 0$, which indicates that s_i need not move if event k appears far away.
3) $f(D_i) - f(D_j) < D_i - D_j, \forall D_i > D_j$, which implies that s_i cannot move past another sensor s_j along the same vector in response to event k.

Two functions can meet the criteria: $f(D_i) = D_i \cdot e^{-D_i}$ and $f(D_i) = \alpha D_i^{\beta} \cdot e^{-\gamma D_i}$ where $\alpha e^{-\gamma D_i} \cdot (\beta D_i^{\beta-1} - \gamma D_i^{\beta}) > 1$ (e.g., $\alpha = 0.06$, $\beta = 3$, and $\gamma = 1$). Besides, the *history-based scheme* lets sensors keep the event history to get a better approximation of event distribution. It uses a *cumulative distribution function (CDF)* of events to update the sensors' positions. Specifically, s_i updates its x-axis position for event k as follows:

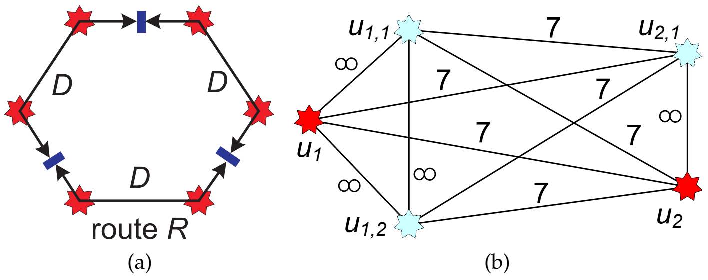


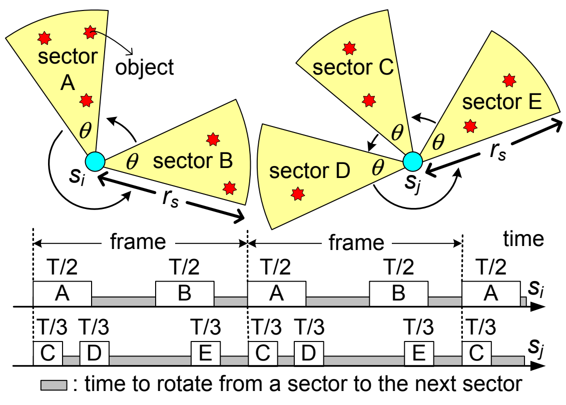
Fig. 8: Sweep coverage: (a) cut route R into three equal pieces and (b) build a weighted complete graph based on two given points u_1 and u_2, where $d(u_1, u_2) = 7$.

1) Increase CDF bins that represent positions $\geq x_k$.
2) Scale CDF by a factor $k/(k + 1)$.
3) Find two CDF bins b_j and b_{j+1} such that $b_j \leq p_i^{k-1}(x) \leq b_{j+1}$, where $p_i^{k-1}(x)$ is the current x-axis position of s_i.
4) Compute the new x-axis position of s_i by interpolating the values of b_j and b_{j+1}.

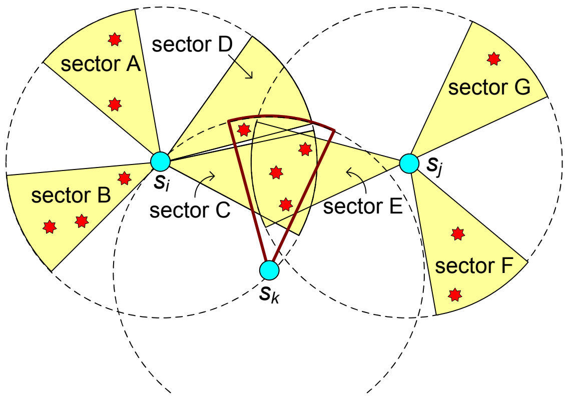
The y-axis position of s_i is updated in the same way.

Given m points of interest, [70] moves sensors to *periodically* cover them. A point u_i is called *t_i-sweep covered* if it is covered by one sensor at least once in a *sweep period* t_i. Then, the *min-sensor sweep coverage problem* decides how to use the minimum sensors to make each point t_i-sweep covered. Let v be a sensor's moving speed. Two schemes are proposed to solve this NP-hard problem. The *CSWEEP scheme* assumes that all points have the same sweep period t and uses a TSP (traveling salesman problem) heuristic [71] to find a route R to visit all points. Then, it cuts R into equal pieces with length of $D = v \cdot t/2$, as shown in Fig. 8(a). Each sensor continues moving along one piece of R back and forth, so each point on the piece is visited by the sensor at least once in every $2 \cdot D/v = t$ period. Besides, the *GSWEEP scheme* allows points to have different sweep periods. For each point u_i, GSWEEP computes its monitoring frequency $f_i = \lceil 1/t_i \rceil$. Let $f_G = gcd(f_1, f_2, \cdots, f_m)$. Then, a weighted complete graph is constructed, where the vertex set contains all m points and the duplications of each point. Here, each point u_i has $k_i = (f_i/f_G) - 1$ duplications. The weight of the edge from a point u_i to another point u_j (or each of u_j's duplications) is $d(u_i, u_j)$. However, the weight is set to infinite for each edge from u_i to its duplication (or between two duplications of u_i). Fig. 8(b) gives an example, where $u_{1,1}$ and $u_{1,2}$ are u_1's duplications, and $u_{2,1}$ is u_2's duplication. The weights of edges $(u_1, u_{1,1})$, $(u_1, u_{1,2})$, $(u_{1,1}, u_{1,2})$, $(u_2, u_{2,1})$ are ∞. GSWEEP uses a TSP heuristic to find a route R to visit all vertices in the graph. It then cuts R into equal pieces, each with length of $D = v/(2f_G)$, and uses one sensor to move along each piece. Since every point u_i has k_i duplications on R, u_i is visited at least $k_i + 1 = f_i/f_G$ times in a $1/f_G$ period. In a t_i period, u_i is visited at least $f_i/f_G \cdot t_i \cdot f_G = 1$ time, so u_i is t_i-sweep covered.

The work of [72] uses *rotatable and directional (R&D) sensors* to cover a set of objects, where the sensing range is a sector with angle of θ and each sensor has the rotation ability. The time axis is divided into *frames* during which a sensor rotates 360° and spends total time T to cover objects. An object is called *δ-time covered* if it is covered by a sensor for at least δT time per frame, where $0 < \delta \leq 1$. Fig. 9(a) gives an example, where sensor s_i divides T equally to cover the objects



(a) system model



(b) DOD method

Fig. 9: Use R&D sensors to cover objects.

in sectors A and B, and sensor s_j divides T equally to cover the objects in sectors C, D, and E in a frame. The objects in sectors $\{A, B\}$ and $\{C, D, E\}$ are 1/2- and 1/3-time covered, respectively. Then, the *R&D sensor deployment problem* decides how to use the minimum R&D sensors to make a set $\mathcal{O}$ of objects δ-time covered. Two methods are designed to solve this NP-hard problem. The *maximum covering deployment (MCD) method* computes a set of disks $\mathcal{D}$ to cover all objects in $\mathcal{O}$ [73]. Then, it uses a greedy strategy to select a disk from $\mathcal{D}$ that covers the maximum objects and then deploys sensor(s) in the disk's center to make its objects δ-time covered. This iteration is repeated until all objects in $\mathcal{O}$ are covered. Besides, the *disk-overlapping deployment (DOD) method* improves the greedy strategy by using disk overlap. When two disks are close enough such that some objects locate inside both disks, these objects may be covered by one sector. Fig. 9(b) shows an example, where $\delta = 1/2$. The MCD method places two sensors at s_i and two sensors at s_j to make all objects $1/2$-time covered, since each sensor supports only two sectors. In fact, the sensor at s_k covers all objects in sectors C, D, and E, so the DOD method requires only three sensors (each at s_i, s_j, and s_k).

4.4 Discussion on Coverage Solutions in Purely Mobile WSNs

Table 5 compares the coverage solutions mentioned in Section 4. For area coverage, the benefit of using virtual forces or a Voronoi diagram is that sensors can move in a *distributed* manner. However, these schemes can only solve the 1-coverage problem since they do not deal with cooperative sensing among sensors. Using geometry to compute the sensor locations helps decide the number of sensors in advance, so it can solve the k-coverage problem [63]. However, the computation has to be done in a centralized manner. [55], [56], [58] assume a *binary sensing model*, where a location is either covered or not covered by a sensor (depending on r_s). This assumption is not practical, so [54], [61], [63] extend their solutions by considering a *probabilistic sensing model*, where a location is covered by a sensor with some probability function. Only [59] addresses *heterogeneous* sensors, where they have different sensing ranges. Such heterogeneity can help improve other coverage solutions since most of them assume that sensors have the same r_s value. Besides, [54], [56], [61] allow the ROI to have obstacles, so their coverage solutions can be applied to practical deployment. However, the current solutions only support coverage in the *space* domain, which means that sensors will never move once they have covered an ROI. Since mobility is the superiority of mobile sensors, one research challenge is how to solve the area coverage problems in the *time* domain, where the ROI(s) may be changed as time goes on. Furthermore, due to the ROI's shape or obstacles, sensors may have irregular sensing range such as polygons [74]. It thus deserves further investigation to use mobile sensors to deal with this case since most existing schemes are based on the assumption of disk-based sensing range.

For barrier coverage, [64] lets sensors form k sensor-disjointed chains to satisfy strong k-barrier coverage, but it only supports coverage in the space domain (i.e., sensors will not move after forming the chains). Motivated by the kinetic theory, [66] finds the k-barrier coverage probability where sensors keep moving. However, sensors move randomly instead of intentionally. [68] intentionally moves sensors to increase the intruder detection probability. However, it only considers the 1-barrier coverage case. Both [66] and [68] support coverage in the time domain since they allow sensors to continue moving to search intruders. However, all of the barrier coverage solutions assume the binary sensing model. Their results can be improved by considering the probabilistic sensing model. In addition, it is critical to detect the moving path of an intruder in a WSN. For example, an intruder's *penetration path* is its crossing path that enters the barrier from one side and leaves the barrier from the other side. Accurately describing the penetration path is important in some monitoring applications. Also, an *exposure path* measures how well a barrier can be covered in terms of the expected ability to detect a moving intruder (the higher the exposure, the better the coverage in the barrier). Both penetration and exposure paths have been studied in static WSNs [75]–[77]. However, it deserves further investigation to use a purely mobile WSN to solve these problems by exploiting sensor mobility.

For point coverage, [9] moves sensors to approximate to the event distribution, so it can use more sensors to detect event occurrence. However, this also involves complicated computation (i.e., complex $f(\cdot)$ function and CDF). Both [70] and [72] formulate NP-hard problems to monitor a set of points/objects in the time domain, which provide a novel perspective on sensor coverage. However, [70] assumes that sensors have the same moving speed while [72] considers that all objects have the equal importance. They can be improved by considering the heterogeneity of sensors and objects. In addition, one could apply the probabilistic sensing model to these coverage solutions to make them more practical. The

TABLE 5: Comparison on the features of different coverage solutions by purely mobile WSNs.

research effort	coverage problem	proposed method	sensing model	coverage domain	obstacles allowed
Area coverage:					
work of [54]	1-coverage	virtual force	probabilistic	space	✓
work of [55]	1-coverage	virtual force	binary	space	
work of [56]	1-coverage	virtual force	binary	space	✓
work of [58]	1-coverage	Voronoi diagram	binary	space	
work of [59]	1-coverage	Voronoi diagram	heterogeneous	space	
work of [61]	1-coverage	geometry	probabilistic	space	✓
work of [63]	k-coverage	geometry	probabilistic	space	
Barrier coverage:					
work of [64]	k-barrier	virtual force	binary	space	
work of [66]	k-barrier	probability	binary	time	
work of [68]	1-barrier	probability	binary	time	
Point coverage:					
work of [9]	point	CDF	binary	time	
work of [70]	point	TSP	binary	time	
work of [72]	point	geometry	sector	time	

topic of point coverage by mobile sensor networks is not well studied yet. One research challenge is to develop various point coverage problems by taking advantage of sensor mobility. Besides, another open research challenge is how to make sensors *cooperate* to satisfy the coverage requirements (e.g., sharing the coverage time of two or more sensors for certain points/objects).

5 Software: Dispatch Algorithms in Hybrid WSNs

A hybrid WSN consists of static and mobile sensors. Static sensors monitor the ROI, whereas mobile sensors tactically move to some locations to conduct certain missions. Sensors can use GPS or other localization schemes [78] to obtain their positions. Following the taxonomy in Fig. 1, this section discusses three kinds of missions by mobile sensors. First, each mobile sensor acts as a data mule or mobile relay to pass data from static sensors to the sink. Second, mobile sensors are responsible for faulty recovery by restoring network coverage or connectivity. Third, mobile sensors visit event locations indicated by static sensors to provide more in-depth analysis.

5.1 Dispatch Mobile Sensors for Data Collection

Mobile sensors can travel in a hybrid WSN to collect data from static sensors. This is useful when the WSN is partitioned into isolated subnetworks. In this case, mobile sensors are also called *data mules* (or *data ferries*) as they collect data on behalf of the sink. Besides, mobile sensors can help relay data for static sensors to save their energy, or even become a part of the routing tree. In this case, mobile sensors are also called *mobile relays*. Below, we introduce the dispatch algorithms to reduce the data delay in a hybrid WSN by exploiting the mobility of data mules and mobile relays.

5.1.1 Data Mules

Given a hybrid WSN containing n data mules, [79] decides their paths to collect data from each static sensor to reduce the average delay. The ROI is divided into n grids. Each grid g_i has a length of l and is handled by a data mule s_i. The TSP method in [80] is applied to compute s_i's path such that the overall data delay in g_i is minimized. Since two data mules in adjacent grids may want to exchange data, two methods are developed. In the *node replying method*, the paths in two grids g_i and g_j can be combined by selecting the static sensor in g_i

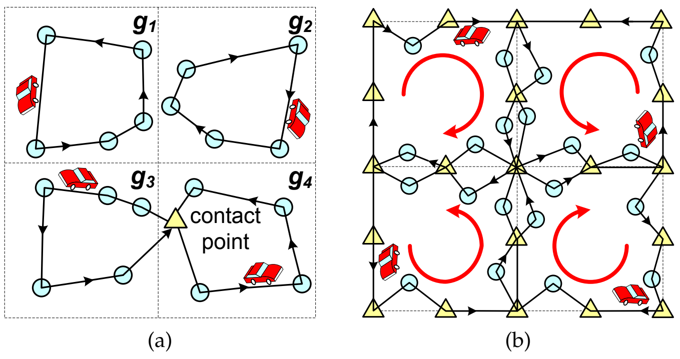
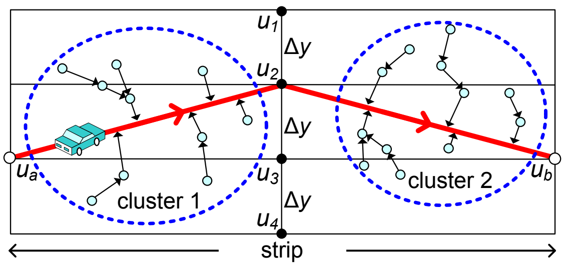


Fig. 10: Move data mules in a grid-based ROI: (a) node replying method and (b) ferry relaying method.

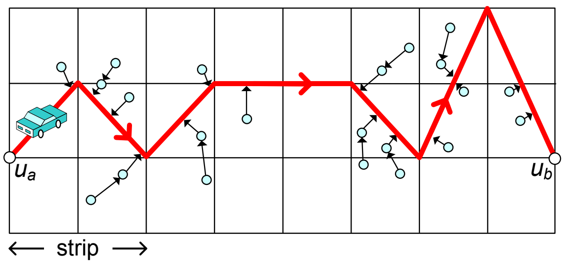
that is closest to g_j as a *contact point*. Then, data mules s_i and s_j meet at the contact point to exchange data. Fig. 10(a) gives an example, where the paths in g_3 and g_4 are combined. Besides, the *ferry relaying method* designates eight contact points along the grid boundary, where any two adjacent contact points are separated by $l/2$. Each data mule s_i computes a path to visit all static sensors and contact points in its grid g_i. The data mules in any two adjacent grids move in reverse directions of each other, so they can meet on some contact points, as shown in Fig. 10(b).

The work of [81] uses a data mule to traverse a static WSN to collect data but the length of its moving path is bounded. Thus, some static sensors cannot meet the data mule and require *multihop relays* to reach it. Since longer relaying paths spend more energy of static sensors, the goal is to compute the moving path of the data mule to save the energy of static sensors (by reducing their relaying paths). The ROI is divided into *strips*, each with width of Δy. Given an entrance point u_a and an exit point u_b, a divide-and-conquer scheme is proposed:

1) Find a *turning point* on the center of each strip's boundary. Then, select the turning point u_i such that when the data mule moves along the path $u_a \Rightarrow u_i \Rightarrow u_b$, static sensors can spend less energy. For example, four turning points u_1, u_2, u_3, u_4 are found in Fig. 11(a) and the moving path $u_a \Rightarrow u_2 \Rightarrow u_b$ is selected.
2) Group static sensors into clusters based on their distances to line segments $\overline{u_a u_i}$ and $\overline{u_i u_b}$, where a static sensor chooses the closest line segment. Then, each strip is cut into two halves accordingly. Static sensors

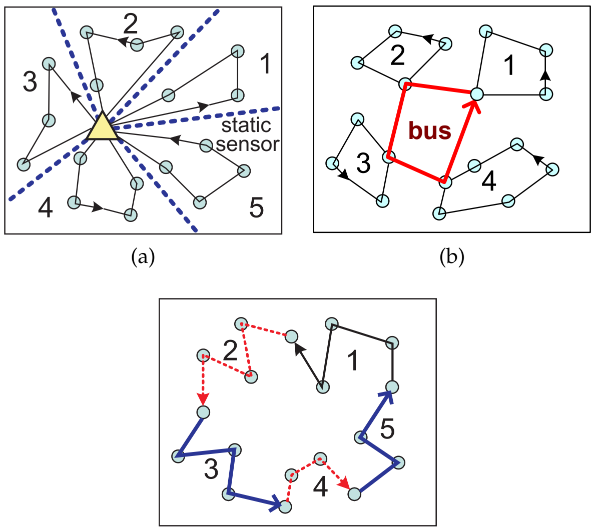


(a) the path in iteration 1



(b) the path in iteration 3

Fig. 11: Divide-and-conquer scheme.



(a) (b) (c)

Fig. 12: Move data mules by Hamiltonian cycles: (a) centralized ferry relaying method, (b) bus ferry relaying method, and (c) neighbor ferry relaying method.

which cannot meet the data mule (along the moving path) should decide the relaying paths to reach it.

3) Repeat the above two steps until the length of the moving path reaches the bound. Fig. 11(b) shows the result after three iterations.

Given a hybrid WSN with n data mules, [82] uses Hamiltonian cycles to decide their paths to visit each static sensor, where a Hamiltonian cycle is a cycle in an undirected graph that visits every node exactly once and returns to the starting node. Three methods are proposed, as shown in Fig. 12. In the *centralized ferry relaying method*, a contact point (marked by a triangle) in the ROI is selected for all data mules to meet and exchange their data. The ROI is divided into n subregions using the contact point as the center. Each subregion is handled by a data mule and a Hamiltonian cycle is formed to visit all static sensors in the subregion and the contact point. Then, the *bus ferry relaying method* forms n Hamiltonian cycles, where $n-1$ cycles are used to visit all static sensors and one special cycle called *bus* is used to connect all other cycles. The data mule on the bus is responsible for exchanging data with others. Finally, the *neighbor ferry relaying method* uses a Hamiltonian cycle to visit all static sensors. Like a relay race, each data mule s_i moves along several segments of the cycle. When s_i meets the next data mule s_{i+1}, it passes data to s_{i+1} and waits for the previous data mule s_{i-1} to pass data again.

5.1.2 Mobile Relays

Given a set of static sensors $\mathcal{S}$ that form a routing tree $\mathcal{T}$ rooted at a sink B, [83] moves a mobile relay along a *cyclic path* to relay data from a subset $\mathcal{S}_v \subseteq \mathcal{S}$ of static sensors to B. The goal is to find the path such that its length $L_p(\mathcal{S}_v) \leq L_{\max}$ and the total hop count from each static sensor to its closest node in $\mathcal{S}_v$ (along $\mathcal{T}$) is minimized. This can alleviate the *funneling effect* [84], where the sensors closer to the sink will exhaust energy faster. Let $\mathcal{S}_v$ be $\{B\}$ initially. Then, a four-step method is developed:

1) Let $\mathcal{W}$ be the candidate set. For each $s_i \in \mathcal{S} - \mathcal{S}_v$, if $L_p(\mathcal{S}_v \cup \{s_i\}) \leq L_{\max}$, s_i is added to $\mathcal{W}$. However, if $\mathcal{W} = \emptyset$ (i.e., we cannot find any s_i), the method terminates.
2) Each $s_i \in \mathcal{W}$ is associated with a *utility* computed by
$$U(s_i) = \frac{\sum_{s_j\in\mathcal{S}} d_{\mathcal{T}}(s_j,\mathcal{S}_v) - \sum_{s_j\in\mathcal{S}} d_{\mathcal{T}}(s_j,\mathcal{S}_v\cup\{s_i\})}{L_p(\mathcal{S}_v\cup\{s_i\}) - L_p(\mathcal{S}_v)},$$
where $d_{\mathcal{T}}(s_j, \mathcal{S}_v)$ is the minimum hop count from s_j to a node in $\mathcal{S}_v$ (along $\mathcal{T}$). Here, the utility of s_i indicates the *reduction ratio* of total hop count that data have to be relayed along $\mathcal{T}$ to the increase of path length after adding s_i. Then, the static sensor s_i with the maximum utility is added to $\mathcal{S}_v$ and removed from $\mathcal{W}$.
3) Recompute the utility of every s_j in $\mathcal{S}_v$. If $U(s_j) = 0$, it is removed from $\mathcal{S}_v$.
4) Repeat the above iteration (i.e., steps 2 and 3) until $\mathcal{W} = \emptyset$.

Fig. 13 gives an example, where $\mathcal{W} = \{s_1, s_2, s_3\}$ by step 1. In iterations 1 and 2, s_1 and s_2 are moved from $\mathcal{W}$ to $\mathcal{S}_v$. In iteration 3, s_3 is added to $\mathcal{S}_v$, resulting in $U(s_1) = 0$. Thus, s_1 is removed from $\mathcal{S}_v$ and the final path is $B \Rightarrow s_2 \Rightarrow s_3 \Rightarrow B$.

The work of [85] uses static and mobile relays to connect multiple isolated WSNs (called *segments*). However, the relays are not enough to connect all segments if they stay stationary. Thus, mobile relays have to traverse between segments to connect them. To decide relay positions, a *minimum Steiner tree*, which is a minimum spanning tree considering extra *Steiner points* to reduce the tree length, is constructed to connect all segments. Each Steiner point is placed with a static relay to serve as a *contact point* for mobile relays. Then, we sort all edges connecting *leaf segments* and select the shortest edge to assign a mobile relay to traverse. This process is repeated until static relays become enough to connect the remaining edges. The rationale is that an edge serving only ingoing/outgoing traffic of a single segment has the least impact on data delay if we use a mobile relay to traverse it. This method can be improved by using just a mobile relay to traverse two adjacent edges if their data delay is below a threshold δ_T. Fig. 14 gives an example, where six segments denoted by L_1 to L_6 are connected by a minimum Steiner tree with three Steiner points S_1, S_2, and S_3. Edges (L_1, S_2), (L_2, S_1), (L_3, S_3), (L_5, S_1), and (L_6, S_3)

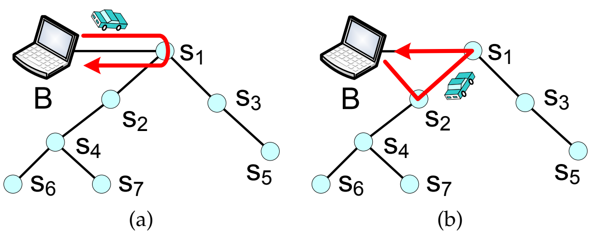


Fig. 13: Form a cyclic path to visit static sensors: (a) iteration 1: $\mathcal{S}_v = \{B\} \cup \{s_1\}$ and $\mathcal{W} = \{s_2, s_3\}$, (b) iteration 2: $\mathcal{S}_v = \{B, s_1\} \cup \{s_2\}$ and $\mathcal{W} = \{s_3\}$, and (c) iteration 3: $\mathcal{S}_v = \{B, s_2\} \cup \{s_3\}$ and $\mathcal{W} = \emptyset$.

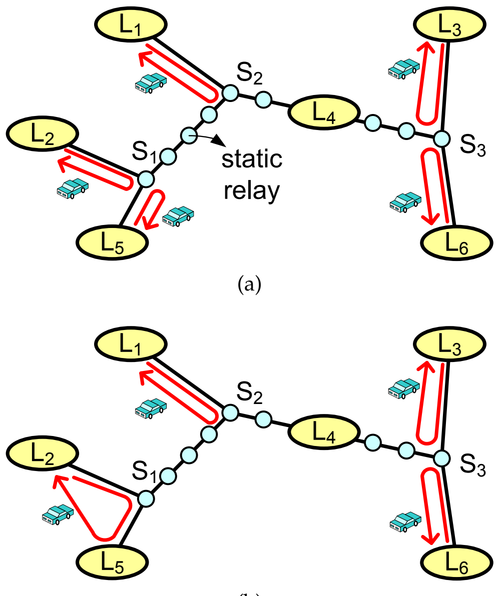


Fig. 14: Use static and mobile relays to connect segments: (a) Construct a minimum Steiner tree and assign relays to connect edges and (b) two edges (L_2, S_1) and (L_5, S_1) are connected by one mobile relay.

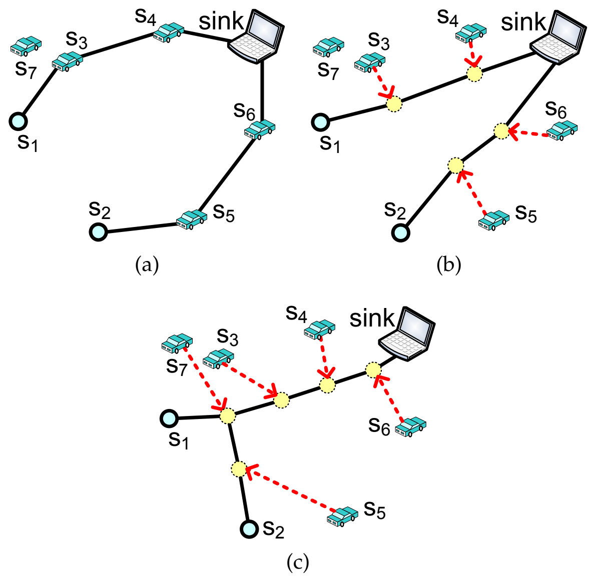


Fig. 15: The change of a routing tree: (a) no mobile sensors should move when m is small, (b) mobile sensors in the tree move when m is medium, and (c) the tree has to include more sensors when m becomes large.

are connected by mobile relays. Other edges are connected by static relays to reduce their data delay. The method is improved by using a mobile relay to traverse both edges (L_2, S_1) and (L_5, S_1) since their data delay keeps below δ_T.

The study of [86] considers a hybrid WSN where static sensors generate CBR (constant bit rate) data and mobile sensors form a network backbone to relay data. The goal is to construct an optimal routing tree from all static sensors to the sink by finding the positions of mobile sensors in the tree such that both the energy spent by relocating mobile sensors and the energy spent by relaying data from static sensors to the sink is minimized. Suppose that a mobile sensor s_i moves from a position o_i to a new position u_i, and it relays m-bit data to the neighbor s_{i+1} at position u_{i+1}. Then, s_i's energy cost is computed by $E(s_i) = (k \cdot d(o_i, u_i)) + (am + bm \cdot d^2(u_i, u_{i+1}))$, where the first and second terms respectively indicate its *sensor-relocation* and *data-relay* energy costs, k depends on the moving speed, and a and b are constants decided by the environment. Then, a three-phase algorithm is proposed to find the optimal tree:

1) Tree construction phase: For any two sensors s_i and s_j at positions o_i and o_j, respectively, their edge weight is defined by $w(s_i, s_j) = a + b \cdot d^2(o_i, o_j)$. Then, we construct a *shortest-path tree* rooted at the sink to all static sensors.
2) Node insertion phase: This tree is improved by adding more relay nodes to save energy. For each mobile sensor s_{out} not in the tree and each tree edge (s_i, s_j), we compute the *reduction* in the energy cost when s_{out} joins the tree such that data is relayed by the new path $s_i \Rightarrow s_{out} \Rightarrow s_j$. Then, we repeatedly insert the mobile sensor s_{out} with the maximum reduction until the tree no longer saves any energy.
3) Tree optimization phase: Some sensors in the tree are relocated to minimize their energy costs $E(s_i)$. The detail can be found in [86].

Fig. 15 gives an example to show how the optimal tree changes, where two static sensors s_1 and s_2 generate CBR data with m bits.

5.2 Dispatch Mobile Sensors for Faulty Recovery

Due to random deployment or node failure, a WSN may have *coverage holes* that are not covered by any sensor. What's worse, the WSN could be *partitioned*. Adding mobile sensors can perform faulty recovery by restoring coverage and connectivity. For instance, [87] proposes a *bidding protocol* to use mobile sensors to fill coverage holes in a hybrid WSN. Each mobile sensor has a *base price* related to the size of any new coverage hole caused by its movement, which indicates the moving cost in terms of coverage. Static sensors use the Voronoi diagram to check if there is a coverage hole in the proximity. If so, a *bid* is estimated by $\pi(D - r_s)^2$, where D is the distance between the bidder (static sensor) and the farthest vertex of the Voronoi polygon. Then, static sensors use their bids to compete for mobile sensors. A mobile sensor only accepts those bids larger than its base price to ensure that its leaving

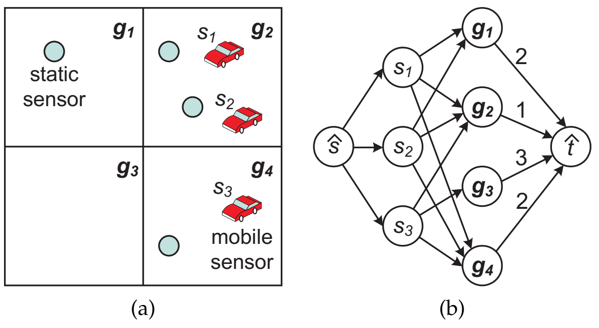


Fig. 16: Move mobile sensors to satisfy desired coverage in each grid: (a) initial WSN deployment and (b) using a flow network to assign mobile sensors, where the capacity of an edge without number is 1.

will not generate a larger coverage hole than that to be filled. It selects the highest bid and moves to fill the largest hole. Then, the mobile sensor's price is updated by that bid. Besides, static sensors use two strategies to select mobile sensors. In the *distance-based strategy*, a bidder selects the closest mobile sensor to bid, which attempts to reduce the average moving distance of mobile sensors. In the *price-based strategy*, a bidder selects the cheapest mobile sensor to bid. Since the selected mobile sensor will generate the smallest coverage hole when it leaves, this strategy could reduce the opportunity that other mobile sensors should move to fill the newly generated hole in the future.

In [88], a hybrid WSN is deployed in an ROI divided into n grids. Each mobile sensor can move in one-grid distance. The *gap* of a grid g_j is defined by $a_j = \max\{k - |g_j|, 0\}$, where k is the desired number of sensors in a grid and $|g_j|$ is the number of sensors in g_j. The goal is to move mobile sensors to minimize three terms: a) the sum of gaps of all grids, b) $(a_1^\alpha + a_2^\alpha + \cdots + a_n^\alpha)^{1/\alpha}$, and c) the total moving cost of mobile sensors. Here, $\alpha = 1$ means minimizing the sum of gaps of all grids, $\alpha = 2$ means minimizing the gap variance, and $\alpha = \infty$ means minimizing the maximum gap. To solve the problem, a bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$ is constructed, where $\mathcal{S}$ is the set of mobile sensors and $\mathcal{L}$ is the set of grids that have fewer than k static sensors. An edge (s_i, g_j), $s_i \in \mathcal{S}, g_j \in \mathcal{L}$ exists if s_i can move to g_j (or it stays in g_j). Then, by adding a source node $\hat{s}$ and a destination node $\hat{t}$, $\mathcal{G}$ becomes a flow network. Each edge is given a *capacity* denoted by $b(\cdot, \cdot)$. For each $s_i \in \mathcal{S}$ and $g_j \in \mathcal{L}$, $b(\hat{s}, s_i) = 1$, $b(s_i, g_j) = 1$, and $b(g_j, \hat{t})$ is the number of mobile sensors required by g_j. Fig. 16 gives an example, where $k = 3$. Grids g_1, g_2, g_3, g_4 have 1, 2, 0, 1 static sensors, so $b(g_1, \hat{t}) = 2, b(g_2, \hat{t}) = 1, b(g_3, \hat{t}) = 3, b(g_4, \hat{t}) = 2$, respectively. Each edge is associated with a *cost* as follows:

1) $c(\hat{s}, s_i) = 0$ and $c(s_i, g_j)$ is the moving cost for s_i to move to g_j.
2) $c(g_j, \hat{t}) = \beta \cdot (b(g_j, \hat{t}) - f(g_j, \hat{t}))^\alpha$, where $\beta = c_{\max} \cdot (|\mathcal{S}|+1)$, $f(g_j, \hat{t})$ is the number of flows on edge $(g_j, \hat{t})$, and $c_{\max}$ is the maximum moving cost of a mobile sensor.

Then, we can find a maximum set of flows from $\hat{s}$ to $\hat{t}$ to minimize the cost function:

$$\sum_{s_i \in \mathcal{S}, g_j \in \mathcal{L}} f(s_i, g_j) \cdot c(s_i, g_j) + \sum_{g_j \in \mathcal{L}} c(g_j, \hat{t}),$$

where $f(s_i, g_j)$ is the number of flows on edge (s_i, g_j), and these flows indicate where to move mobile sensors.

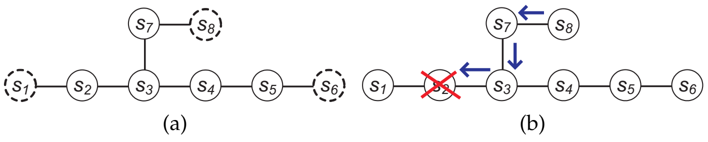


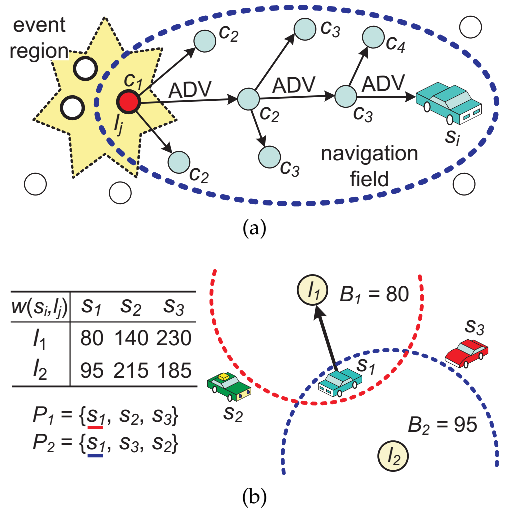
Fig. 17: Deal with network partition by mobile sensors: (a) initial network topology, where dominatees are marked by dashed circles and (b) cascaded movement when s_2 fails.

The work of [89] deals with network partition due to node failure by relocating mobile sensors. A WSN must be partitioned if any of its *cut-vertices* fails. Thus, each sensor s_i checks if it is a cut-vertex. To do so, s_i exchanges data with its two-hop neighbors to know whether it is a *dominator* (a term in the *connected dominating set, CDS* [90]) or a *dominatee* one-hop away from a dominator. If s_i is a dominatee, it cannot be a cut-vertex. Otherwise, s_i uses a depth-first search tree to check if there is an alternative path for each of its neighbors to reach the network when s_i fails. If not, s_i is a cut-vertex. Fig. 17(a) shows an example, where s_1 is not a cut-vertex since it is a dominatee and s_2 is a cut-vertex since it is a dominator and when it fails, s_1 cannot find any path to reach the network. Each cut-vertex selects one neighbor as its *failure handler*. The failure handler periodically checks if its cut-vertex is alive by exchanging *heart-beat packets*. Once the cut-vertex fails, its failure handler asks the nearest dominatee to replace the cut-vertex. Since the dominatee may be far away, which wastes its energy on movement, a *cascaded movement method* shown in Fig. 17(b) is used to balance the moving costs of mobile sensors. When the cut-vertex s_2 fails, its failure handler s_3 asks the nearest dominatee s_8 to replace s_2. However, s_8 does not directly move to replace s_2. Instead, s_3 moves to replace s_2, s_7 moves to replace s_3, and s_8 moves to replace s_7. Thus, s_8 can save its energy on movement.

5.3 Dispatch Mobile Sensors for Event Analysis

Several studies use static and mobile sensors in a hybrid WSN to collaboratively monitor events. Static sensors notify mobile sensors of event occurrence. Mobile sensors then move to the *event locations* to perform in-depth analysis. In [91], static sensors detecting events will invite mobile sensors to visit them for further analysis. Since sensors do not know their positions, static sensors have to navigate mobile sensors. Specifically, static sensors detecting the same event elect a leader l_j to broadcast a *weight request packet* to find mobile sensors. On receiving this packet, a mobile sensor s_i responds a weight $A(s_i) \cdot d(s_i, l_j)/E_i$, where $A(s_i)$ is the size of the coverage hole when s_i leaves the current position and it is estimated by the area of s_i's Voronoi polygon, $d(s_i, l_j)$ is the hop count between s_i and l_j, and E_i is s_i's residual energy. After collecting weight responses, l_j selects the mobile sensor s_i with the minimum weight. In case of a tie, the mobile sensor with the minimum $d(s_i, l_j)$ is selected. Then, l_j builds a *navigation field* to guide s_i by sending *advertisement (ADV)* packets along the path from l_j to s_i. In particular, l_j sets the highest credit value c_1 for itself. Then, for each rebroadcast of ADV, a lower credit value is set. Fig. 18(a) gives an example, where $c_1 > c_2 > c_3 > c_4$. Then, s_i moves toward l_j by searching higher credit values.

The work of [92] formulates a *multi-round sensor dispatch problem*, which is NP-complete. Given a set of mobile sensors $\mathcal{S}$ and a set of event locations $\mathcal{L}$ reported by static sensors in each round, this problem asks to dispatch a mobile sensor



$w(s_i,l_j)$	s_1	s_2	s_3
l_1	80	140	230
l_2	95	215	185

$P_1 = \{s_1, s_2, s_3\}$
$P_2 = \{s_1, s_3, s_2\}$

Fig. 18: Examples of navigating and dispatching mobile sensors to event locations: (a) leader l_j builds a navigation field by credit values and (b) event locations compete for mobile sensors by bounds.

to visit each event location so that the lifetime (in rounds) is maximized. When $|\mathcal{L}| > |\mathcal{S}|$, event locations are grouped in advance so that each group is visited by one mobile sensor. Then, a heuristic to calculate a *one-to-one assignment* $\mathcal{M}$ between $\mathcal{L}$ and $\mathcal{S}$ is developed, whose goal is to balance and reduce the moving costs of mobile sensors:

1) Let $w(s_i, l_j)$ be the energy cost for a mobile sensor s_i to visit an event location l_j. Then, each $l_j \in \mathcal{L}$ is given a *preference list* P_j that contains mobile sensors ranked by their $w(s_i, l_j)$ values in an increasing order. Besides, l_j is given a bound B_j to limit the mobile sensors that it can select. The bound B_j is initially set to $w(s_i, l_j)$ such that s_i is the αth element in P_j.
2) Each $l_j \in \mathcal{L}$ selects its first element s_i from P_j. If s_i is unassigned, (s_i, l_j) is added to $\mathcal{M}$. Otherwise, $\mathcal{M}$ must have a pair (s_i, l_o), so l_j and l_o bid for s_i. Three conditions let l_j win the bid: a) $B_j > B_o$, b) $B_j = B_o$ and $w(s_i, l_j) < w(s_i, l_o)$, and c) $B_j = B_o$ and s_i is the only candidate of l_j but not of l_o. If so, (s_i, l_o) is replaced by (s_i, l_j) in $\mathcal{M}$. Otherwise, s_i is removed from P_j and l_j checks for other candidate(s).
3) If l_j does not have any candidate under bound B_j, it increases B_j to $w(s_k, l_j)$ such that s_k is the αth element in P_j and goes back to step 2.
4) Repeat steps 2 and 3 until all locations in $\mathcal{L}$ are assigned with mobile sensors.

Here, the bounds are used to balance the energy spent by mobile sensors to visit event locations. Fig. 18(b) shows an example, where $\alpha = 1$. Initially, $\mathcal{M} = \{(s_1, l_1)\}$. Then, l_2 and l_1 bid for s_1. Since $B_2 > B_1$, l_2 wins the bid. So, (s_1, l_1) is replaced by (s_1, l_2) in $\mathcal{M}$. In this case, s_1 is removed from P_1 and l_1 no longer has candidates. Thus, l_1 increases its bound to $w(s_2, l_1) = 140$ and selects s_2. The final result is $\mathcal{M} = \{(s_1, l_2), (s_2, l_1)\}$.

The study of [93] extends the multi-round sensor dispatch problem by considering *multi-attribute* mobile sensors. Each event reported by static sensors has one attribute but a mobile sensor can analyze multiple attributes of events. Mobile sensors may have different attributes, so only the mobile sensors with the *correct* attribute can analyze a certain event. This problem is NP-complete and a two-phase heuristic is proposed. In phase 1, a weighted bipartite graph $\mathcal{G} = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$ is constructed, where all mobile sensors in $\mathcal{S}$ and all event locations in $\mathcal{L}$ are converted into vertices, and edges connect vertices between $\mathcal{S}$ and $\mathcal{L}$. For each $s_i \in \mathcal{S}$ and each $l_j \in \mathcal{L}$, an edge (s_i, l_j) exists if both s_i and l_j have the same attribute and it is given a weight $w(s_i, l_j)$. Then, the goal is to find a *maximum Pareto-optimal matching* $\mathcal{M}$ from $\mathcal{G}$. Specifically, given two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$, a mobile sensor s_i prefers $\mathcal{M}_1$ to $\mathcal{M}_2$ if either condition is true:

1) Mobile sensor s_i is matched to an event location in $\mathcal{M}_1$ but not in $\mathcal{M}_2$.
2) Assume that s_i is matched to event locations l_j and l_k in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. It spends less energy to reach l_j than to l_k, that is, $w(s_i, l_j) < w(s_i, l_k)$.

Let us denote by $\mathcal{M}_1 >_p \mathcal{M}_2$ if no mobile sensors prefer $\mathcal{M}_2$ to $\mathcal{M}_1$ and some mobile sensors prefer $\mathcal{M}_1$ to $\mathcal{M}_2$. A matching $\mathcal{M}$ is called *Pareto optimal* if there is no other matching $\mathcal{M}'$ such that $\mathcal{M}' >_p \mathcal{M}$. In [94], a scheme to find $\mathcal{M}$ is developed, which takes $O(|\mathcal{S}| \cdot |\mathcal{L}| \cdot \sqrt{|\mathcal{S}| + |\mathcal{L}|})$ time. Then, phase 2 deals with the event locations not in $\mathcal{M}$ by growing spanning trees. Each pair (s_i, l_j) in $\mathcal{M}$ is viewed as a spanning tree rooted at s_i. For each $l_k \in \mathcal{L}$ not in $\mathcal{M}$, l_k joins a spanning tree such that a) its mobile sensor has the correct attribute, b) the tree weight is the minimum, and c) after l_k joins the tree, the increase in the tree weight is minimized. Thus, each spanning tree has a smaller and similar weight, so mobile sensors can save their energy and balance moving costs when visiting event locations.

5.4 Discussion on Dispatch Algorithms in Hybrid WSNs

Table 6 compares the dispatch algorithms addressed in Section 5. Most data collection problems in a hybrid WSN (except [86]) are NP-complete or NP-hard. [79], [81] divide the ROI into grids while [83], [85], [86] organize a tree structure to solve their problems. However, both [81] and [83] aim at scheduling the path of one single mobile sensor, whereas other studies use multiple mobile sensors to cooperatively collect/relay data, which are more efficient. Only [86] proposes a distributed solution and it allows the mobile sensors to dynamically change the positions when the WSN generates different amount of data. Thus, it is more practical for a large-scale WSN (since sensors can execute the solution by themselves). One interesting research challenge arises when there exist *wireless chargers* [95] in the ROI. Mobile sensors can approach these chargers to recharge their batteries but this will spend some time. Thus, a mobile sensor need not meet a charger if it has sufficient energy. In this case, it will be a challenging task to adaptively schedule the moving paths of mobile sensors by visiting *all* necessary locations (e.g., static sensors) and *some* wireless chargers such that they can have sufficient energy to complete their jobs *on time*. Besides, in-network data reduction technique such as *serial data fusion* [96] can significantly reduce the amount of data transmission. Therefore, another research challenge is how to schedule the paths of mobile sensors by exploiting this technique so that they can efficiently collect data from the static sensors.

For faulty recovery, both [87] and [88] aim at restoring a WSN's coverage. However, [87] deals with the problem in a randomly deployed WSN, whereas [88] divides the ROI into grids in advance and tries to distribute mobile sensors over

TABLE 6: Comparison on the features of different dispatch algorithms in hybrid WSNs.

research effort	proposed method	network structure	NP hardness	distributed solution	fixed path
Data collection:					
work of [79]	TSP	grid	✓		✓
work of [81]	divide & conquer	grid	✓		✓
work of [82]	Hamiltonian cycle	random	✓		✓
work of [83]	tree traversal	tree	✓		✓
work of [85]	Steiner tree	tree	✓		✓
work of [86]	routing tree	tree		✓	
Faulty recovery:					
work of [87]	bidding	random	✓	✓	
work of [88]	flow network	grid			✓
work of [89]	CDS	random		✓	
Event analysis:					
work of [91]	bidding	random		✓	
work of [92]	bidding	random	✓	✓	
work of [93]	matching	random	✓		

the grids to let each grid have enough sensors (so the paths of mobile sensors are fixed). Thus, [87] is more efficient than [88] since mobile sensors can *dynamically* move to heal coverage holes. Besides, [87], [89] propose distributed solutions for faulty recovery, which are practical for WSN applications.

For event analysis, both [92] and [93] address the multi-round sensor dispatch problem, which is NP-complete. However, [92] can be viewed as a special case of [93] by assuming that all mobile sensors and events have the same (and only one) attribute. Thus, [93] is more efficient than [92] because it addresses a more generalized problem. In addition, [91], [92] are more practical since they develop distributed solutions to dispatch mobile sensors. Notice that when mobile sensors are dispatched for faulty recovery or event analysis in a *distributed* manner, mobile sensors in fact adopt a *competitive* strategy to contend for the faulty/event locations. Therefore, it deserves further investigation to apply the *game theory* [97] to dispatching mobile sensors in a hybrid WSN.

6 Conclusion

Comparing with static WSNs, mobile sensor networks have more flexibility and capability to cope with variable network conditions and environmental situations. This article gives a complete picture of the research progress in mobile sensors network by presenting a comprehensive survey of their system hardware and dispatch software. We introduce two popular types of mobile sensor hardware. One is to develop mobile robots in which sensors can embed, whereas the other is to use existing conveyances to carry sensors. They have different locomotion models, navigation and localization schemes, communication styles, and energy-conservation policies. Thus, mobile sensors can be applied to various environments and applications. According to the categories of mobile sensor networks, our discussion on dispatch software is separated into two aspects. First, we show how to solve different coverage problems by using a purely mobile WSN, where all sensors are capable of moving. They can adaptively adjust network topology to satisfy coverage requirements including area coverage, barrier coverage, and point coverage. Then, in a hybrid WSN, we discuss how mobile sensors can cooperate with static sensors to complete various missions such as data collection, faulty recovery, and event analysis. A number of open research challenges arise when sensors are possessed of mobility, which have also been addressed in the article.

References

[1] A. Boukerche, *Algorithms and Protocols for Wireless Sensor Networks*, Wiley-IEEE Press, 2008.

[2] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*, Wiley, 2010.

[3] S.S. Dhillon and K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks," *Proc. IEEE Wireless Comm. and Networking Conf.*, 2003, pp. 1609–1614.

[4] Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Trans. Computers*, vol. 54, no. 8, pp. 978–991, 2005.

[5] C.F. Huang, L.C. Lo, Y.C. Tseng, and W.T. Chen, "Decentralized energy-conserving and coverage-preserving protocols for wireless sensor networks," *ACM Trans. Sensor Networks*, vol. 2, no. 2, pp. 182–187, 2006.

[6] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, "Localized sensor area coverage with low communication overhead," *IEEE Trans. Mobile Computing*, vol. 7, no. 5, pp. 661–672, 2008.

[7] Y.C. Wang, F.J. Wu, and Y.C. Tseng, "Mobility management algorithms and applications for mobile sensor networks," *Wireless Comm. and Mobile Computing*, vol. 12, no. 1, pp. 7–21, 2012.

[8] G. Cao, G. Kesidis, T.F.L. Porta, B. Yao, and S. Phoha, *Sensor Network Operations*, Wiley-IEEE Press, 2006, name of chapter: Purposeful mobility in tactical sensor networks.

[9] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34–42, 2003.

[10] 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.

[11] 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*, 2007, pp. 669–674.

[12] A.A. Abbasi, M. Younis, and K. Akkaya, "Movement-assisted connectivity restoration in wireless sensor and actor networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 20, no. 9, pp. 1366–1379, 2009.

[13] K.W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 8, pp. 929–942, 2007.

[14] Y.C. Wang, Y.Y. Hsieh, and Y.C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Trans. Computers*, vol. 58, no. 6, pp. 827–838, 2009.

[15] E. Hyytia, P. Lassila, and J. Virtamo, "Spatial node distribution of the random waypoint mobility model with applications," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 680–694, 2006.

[16] H. Hartenstein and K.P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Comm. Magazine*, vol. 46, no. 6, pp. 164–171, 2008.

[17] S. Panichpapiboon and W. Pattara-Atikom, "A review of information dissemination protocols for vehicular ad hoc networks," *IEEE Comm. Surveys & Tutorials*, vol. 14, no. 3, pp. 784–798, 2012.

[18] Y.C. Wang, C.C. Hu, and Y.C. Tseng, "Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks," *Proc. IEEE Wireless Internet Conf.*, 2005, pp. 114–121.

[19] S. Basagni, A. Carosi, and C. Petrioli, *Algorithms and Protocols for Wireless Sensor Networks*, Wiley-IEEE Press, 2008, name of chapter: Mobility in wireless sensor networks.

[20] M.D. Francesco, S.K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: a survey," *ACM Trans. Sensor Networks*, vol. 8, no. 1, pp. 7:1–7:31, 2011.

[21] Q. Dong and W. Dargie, "A survey on mobility and mobility-aware MAC protocols in wireless sensor networks," *IEEE Comm. Surveys & Tutorials*, vol. 15, no. 1, pp. 88–100, 2013.

[22] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G.S. Sukhatme, "Robomote: enabling mobility in sensor networks," *Proc. ACM Int'l Symp. Information Processing in Sensor Networks*, 2005, pp. 404–409.

[23] D. Cruz, J. McClintock, B. Perteet, O.A.A. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control: a multivehicle platform for research in networked embedded systems," *IEEE Control Systems*, vol. 27, no. 3, pp. 58–78, 2007.

[24] M. Friedrich, G. Dobie, C.C. Chan, S.G. Pierce, W. Galbraith, S. Marshall, and G. Hayward, "Miniature mobile sensor platforms for condition monitoring of structures," *IEEE Sensors J.*, vol. 9, no. 11, pp. 1439–1448, 2009.

[25] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking moving devices with the Cricket location system," *Proc. ACM Int'l Conf. Mobile Systems, Applications, and Services*, 2004, pp. 190–202.

[26] S. Biswas, S. Gupta, F. Yu, and T. Wu, "A networked mobile sensor test-bed for collaborative multi-target tracking applications," *Wireless Networks*, vol. 16, no. 5, pp. 1329–1344, 2010.

[27] 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.

[28] T.A. Riggs, T. Inanc, and W. Zhang, "An autonomous mobile robotics testbed: construction, validation, and experiments," *IEEE Trans. Control Systems Technology*, vol. 18, no. 3, pp. 757–766, 2010.

[29] J.H. Cui, J. Kong, M. Gerla, and S. Zhou, "The challenges of building scalable mobile underwater wireless sensor networks for aquatic applications," *IEEE Network*, vol. 20, no. 3, pp. 12–18, 2006.

[30] Y. Hu, W. Zhao, and L. Wang, "Vision-based target tracking and collision avoidance for two autonomous robotic fish," *IEEE Trans. Industrial Electronics*, vol. 56, no. 5, pp. 1401–1410, 2009.

[31] S. Shatara and X. Tan, "An efficient, time-of-flight-based underwater acoustic ranging system for small robotic fish," *IEEE J. Oceanic Engineering*, vol. 35, no. 4, pp. 837–846, 2010.

[32] X. Tan, "Autonomous robotic fish as mobile sensor platforms: challenges and potential solutions," *Marine Technology Society J.*, vol. 45, no. 4, pp. 31–40, 2011.

[33] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: safe and efficient load manipulation with aerial robots," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 69–79, 2012.

[34] B. Kate, J. Waterman, K. Dantu, and M. Welsh, "Simbeeotic: a simulator and testbed for micro-aerial vehicle swarm experiments," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks*, 2012, pp. 49–60.

[35] A. Purohit, Z. Sun, and P. Zhang, "SugarMap: location-less coverage for micro-aerial sensing swarms," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks*, 2013, pp. 253–264.

[36] P. Zhang, C.M. Sadler, S.A. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, 2004, pp. 227–238.

[37] M.O. Monod, P. Faure, L. Moiroux, and P. Rameau, "A virtual fence for animals management in rangelands," *Proc. IEEE Mediterranean Electrotechnical Conf.*, 2008, pp. 337–342.

[38] V. Dyo, S.A. Ellwood, D.W. Macdonald, A. Markham, N. Trigoni, R. Wohlers, C. Mascolo, B. Pasztor, S. Scellato, and K. Yousef, "WILDSENSING: design and deployment of a sustainable sensor network for wildlife monitoring," *ACM Trans. Sensor Networks*, vol. 8, no. 4, pp. 29:1–29:33, 2012.

[39] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, 2006, pp. 125–138.

[40] S.B. Eisenman, E. Miluzzo, N.D. Lane, R.A. Peterson, G.S. Ahn, and A.T. Campbell, "BikeNet: a mobile sensing system for cyclist experience mapping," *ACM Trans. Sensor Networks*, vol. 6, no. 1, pp. 6:1–6:39, 2009.

[41] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *IEEE Trans. Vehicular Technology*, vol. 58, no. 2, pp. 882–901, 2009.

[42] S.C. Hu, Y.C. Wang, C.Y. Huang, and Y.C. Tseng, "Measuring air quality in city areas by vehicular wireless sensor networks," *J. Systems and Software*, vol. 84, no. 11, pp. 2005–2012, 2011.

[43] P. Lytrivis, G. Thomaidis, M. Tsogas, and A. Amditis, "An advanced cooperative path prediction algorithm for safety applications in vehicular networks," *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 3, pp. 669–679, 2011.

[44] E.A. Wan and R. van der Merwe, *Kalman Filtering and Neural Networks*, Wiley, 2002, name of chapter: The unscented Kalman filter.

[45] R.N. Smith, J. Das, H. Heidarsson, A.M. Pereira, F. Arrichiello, I. Cetnic, L. Darjany, M.E. Garneau, M.D. Howard, C. Oberg, M. Ragan, E. Seubert, E.C. Smith, B.A. Stauffer, A. Schnetzer, G. Toro-Farmer, D.A. Caron, B.H. Jones, and G.S. Sukhatme, "USC CINAPS builds bridges: observing and monitoring the Southern California Bight," *IEEE Robotics & Automation Magazine*, vol. 17, no. 1, pp. 20–30, 2010.

[46] B.M. Howe, Y. Chao, P. Arabshahi, S. Roy, T. McGinnis, and A. Gray, "A smart sensor web for ocean observation: fixed and mobile platforms, integrated acoustics, satellites and predictive modeling," *IEEE J. Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 3, no. 4, pp. 507–521, 2010.

[47] J.F. Vasconcelos, C. Silvestre, and P. Oliveira, "INS/GPS aided by frequency contents of vector observations with application to autonomous surface crafts," *IEEE J. Oceanic Engineering*, vol. 36, no. 2, pp. 347–363, 2011.

[48] W. Gao, G. Cao, T.L. Porta, and J. Han, "On exploiting transient social contact patterns for data forwarding in delay-tolerant networks," *IEEE Trans. Mobile Computing*, vol. 12, no. 1, pp. 151–165, 2013.

[49] S. Zeadally, R. Hunt, Y.S. Chen, A. Irwin, and A. Hassan, "Vehicular ad hoc networks (VANETS): status, results, and challenges," *Telecomm. Systems*, vol. 50, no. 4, pp. 217–241, 2012.

[50] J.W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, 2004, pp. 81–94.

[51] B. Wang, "Coverage problems in sensor networks: a survey," *ACM Computing Surveys*, vol. 43, no. 4, pp. 32:1–32:53, 2011.

[52] S. Kumar, T.H. Lai, and A. Arora, "Barrier coverage with wireless sensors," *Proc. ACM Int'l Conf. Mobile Computing and Networking*, 2005, pp. 284–298.

[53] Y.C. Wang, K.Y. Cheng, and Y.C. Tseng, "Using event detection latency to evaluate the coverage of a wireless sensor network," *Computer Comm.*, vol. 30, no. 14-15, pp. 2699–2707, 2007.

[54] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," *ProcIEEE INFOCOM*, 2003, pp. 1293–1303.

[55] N. Heo and P.K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *IEEE Trans. Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 78–92, 2005.

[56] H. Liu, X. Chu, Y.W. Leung, and R. Du, "Simple movement control algorithm for bi-connectivity in robotic sensor networks," *IEEE J. Selected Areas in Comm.*, vol. 28, no. 7, pp. 994–1005, 2010.

[57] F. Aurenhammer, "Voronoi diagrams–a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[58] 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.

[59] N. Bartolini, T. Calamoneri, T.F.L. Porta, and S. Silvestri, "Autonomous deployment of heterogeneous mobile sensors," *IEEE Trans. Mobile Computing*, vol. 10, no. 6, pp. 753–766, 2011.

[60] H. Imai, M. Iri, and K. Murota, "Voronoi diagram in the Laguerre geometry and its applications," *SIAM J. Computing*, vol. 14, no. 1, pp. 93–105, 1985.

[61] 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.

[62] H.W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 2, no. 1–2, pp. 83–97, 1955.

[63] 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.

[64] L. Kong, X. Liu, Z. Li, and M.Y. Wu, "Automatic barrier coverage formation with mobile sensor networks," *Proc. IEEE Int'l Conf. Comm.*, 2010, pp. 1–5.

[65] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Comm. and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.

[66] G.Y. Keung, B. Li, and Q. Zhang, "The intrusion detection in mobile sensor network," *IEEE/ACM Trans. Networking*, vol. 20, no. 4, pp. 1152–1161, 2012.

[67] R.D. Present, *Kinetic Theory of Gases*, McGraw-Hill, 1958.

[68] S. He, J. Chen, X. Li, X. Shen, and Y. Sun, "Cost-effective barrier coverage by mobile sensor networks," *Proc. IEEE INFOCOM*, 2012, pp. 819–827.

[69] W. Weibull, “A statistical distribution function of wide applicability,” *J. Applied Mechanics*, vol. 18, no. 3, pp. 293–297, 1951.

[70] M. Li, W. Cheng, K. Liu, Y. He, X. Li, and X. Liao, “Sweep coverage with mobile sensors,” *IEEE Trans. Mobile Computing*, vol. 10, no. 11, pp. 1534–1545, 2011.

[71] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics, 2007.

[72] Y.C. Wang, Y.F. Chen, and Y.C. Tseng, “Using rotatable and directional (R&D) sensors to achieve temporal coverage of objects and its surveillance application,” *IEEE Trans. Mobile Computing*, vol. 11, no. 8, pp. 1358–1371, 2012.

[73] B. Xiao, Q. Zhuge, Y. He, Z. Shao, and E.H.M. Sha, “Algorithms for disk covering problems with the most points,” *Proc. IASTED Int'l Conf. Parallel and Distributed Computing and Systems*, 2003, pp. 541–546.

[74] A. Boukerche and X. Fei, “A coverage-preserving scheme for wireless sensor network with irregular sensing range,” *Ad Hoc Networks*, vol. 5, no. 8, pp. 1303–1316, 2007.

[75] S. Megerian, F. Koushanfar, G. Qu, G. Veltri, and M. Potkonjak, “Exposure in wireless sensor networks: theory and practical solutions,” *Wireless Networks*, vol. 8, no. 5, pp. 443–454, 2002.

[76] S. Megerian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, “Worst and best-case coverage in sensor networks,” *IEEE Trans. Mobile Computing*, vol. 4, no. 1, pp. 84–92, 2005.

[77] Y.T. Hou, C.M. Chen, and B. Jeng, “An optimal new-node placement to enhance the coverage of wireless sensor networks,” *Wireless Networks*, vol. 16, no. 4, pp. 1033–1043, 2010.

[78] A. Boukerche, H.A.B. Oliveira, E.F. Nakamura, and A.A.F. Loureiro, “Localization systems for wireless sensor networks,” *IEEE Wireless Comm.*, vol. 14, no. 6, pp. 6–12, 2007.

[79] W. Zhao, M. Ammar, and E. Zegura, “Controlling the mobility of multiple data transport ferries in a delay-tolerant network,” *Proc. IEEE INFOCOM*, 2005, pp. 1407–1418.

[80] W. Zhao and M.H. Ammar, “Message ferrying: proactive routing in highly-partitioned wireless ad hoc networks,” *Proc. IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003, pp. 308–314.

[81] 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.

[82] R. Moazzez-Estanjini, J. Wang, and I.C. Paschalidis, “Scheduling mobile nodes for cooperative data transport in sensor networks,” *IEEE/ACM Trans. Networking*, vol. 21, no. 3, pp. 974–989, 2013.

[83] 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.

[84] G.S. Ahn, S.G. Hong, E. Miluzzo, A.T. Campbell, and F. Cuomo, “Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks,” *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, 2006, pp. 293–306.

[85] J.L.V.M. Stanislaus and M. Younis, “Delay-conscious federation of multiple wireless sensor network segments using mobile relays,” *Proc. IEEE Vehicular Technology Conf.*, 2012, pp. 1–5.

[86] F. El-Moukaddem, E. Torng, and G. Xing, “Mobile relay configuration in data-intensive wireless sensor networks,” *IEEE Trans. Mobile Computing*, vol. 12, no. 2, pp. 261–273, 2013.

[87] 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. 563–576, 2007.

[88] Z. Shen, Y. Chang, H. Jiang, Y. Wang, and Z. Yan, “A generic framework for optimal mobile sensor redeployment,” *IEEE Trans. Vehicular Technology*, vol. 59, no. 8, pp. 4043–4057, 2010.

[89] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, “Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility,” *IEEE Trans. Computers*, vol. 59, no. 2, pp. 258–271, 2010.

[90] F.V. Fomin and D.M. Thilikos, “Dominating sets in planar graphs: branch-width and exponential speed-up,” *SIAM J. Computing*, vol. 36, no. 2, pp. 281–309, 2006.

[91] 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.

[92] Y.C. Wang, W.C. Peng, and Y.C. Tseng, “Energy-balanced dispatch of mobile sensors in a hybrid wireless sensor network,” *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 12, pp. 1836–1850, 2010.

[93] Y.C. Wang, “A two-phase dispatch heuristic to schedule the movement of multi-attribute mobile sensors in a hybrid wireless sensor network,” *IEEE Trans. Mobile Computing*, vol. 13, no. 4, pp. 709–722, 2014.

[94] D.J. Abraham, K. Cechlarova, D.F. Manlove, and K. Mehlhorn, “Pareto optimality in house allocation problems,” *Proc. Int'l Symp. Algorithms and Computation*, 2005, pp. 1163–1175.

[95] V. Boscaino, F. Pellitteri, L. Rosa, and G. Capponi, “Wireless battery chargers for portable applications: design and test of a high-efficiency power receiver,” *IET Power Electronics*, vol. 6, no. 1, pp. 20–29, 2013.

[96] A. Mostefaoui, M. Melkemi, and A. Boukerche, “Efficient algorithm for serial data fusion in wireless sensor networks,” *Proc. ACM Int'l Conf. Modeling, Analysis & Simulation of Wireless and Mobile Systems*, 2013, pp. 181–188.

[97] S. Tadelis, *The Game Theory: An Introduction*, Princeton University Press, 2013.