Using Rotatable and Directional (R&D) Sensors to Achieve Temporal Coverage of Objects and Its Surveillance Application

You-Chiun Wang, *Member*, *IEEE*, Yung-Fu Chen, and
Yu-Chee Tseng, *Senior Member*, *IEEE*

Abstract—Due to hardware design or cost consideration, sensors may possess sector-like sensing coverage. Furthermore, by stepper motors, sensors can rotate to cover the objects around them. This type of sensors are called *rotatable and directional (R&D) sensors*. Through rotation, R&D sensors provide *temporal coverage* to objects by "periodically" detecting their existence. In the paper, we first develop an event-driven surveillance system by R&D sensors, where objects are monitored by the sensors equipped with infrared detectors and cameras. When an object is taken away, the sensor monitoring the object reports a warning message along with detailed snapshots from the surroundings. Then, motivated by the system, we formulate an *R&D sensor deployment problem*, which tries to deploy the minimum number of R&D sensors to cover a given set of objects such that each object is covered by $0 < \delta \leq 1$ ratio of time in every frame. We show this problem to be NP-hard and propose two efficient heuristics. The *maximum covering deployment (MCD)* heuristic iteratively deploys a sensor to cover more objects, and performs well when objects congregate together. The *disk-overlapping deployment (DOD)* heuristic deploys sensors to cover the joint sectors of overlapped disks, so it works better when objects are arbitrarily placed in the sensing field. The paper contributes in defining a new temporal coverage model by R&D sensors, developing a surveillance application for this model, and proposing efficient heuristics to reduce the deployment cost.

Index Terms—Directional sensor, surveillance system, temporal coverage, wireless sensor network.

1 Introduction

Wireless sensor networks (WSNs) are characterized by ad hoc networking, cooperative sensing, and distributed processing. They are widely adopted in various military and civil applications [1]. Sensors need to organize a connected network that covers a sensing field or specific point-locations to make the WSN function well. Conventional research on WSNs usually assumes omnidirectional sensors with disk-like sensing coverage [2]. However, due to hardware design or cost consideration, sensors may possess sector-like sensing coverage, which we call *directional sensors*. Practical examples include infrared, camera, and ultrasonic sensors. By integrating directional sensors with robotic actuators, such as stepper motors, these sensors can rotate to provide spatiotemporal monitoring of the environment [3]. We call the sensors with such capability *rotatable and directional (R&D) sensors*.

R&D sensors have many real-life applications, for example, providing visual monitoring of the environment [4], [5] and identifying the positions of objects [6]. In the paper, we first prototype an event-driven surveillance system by R&D sensors, where each sensor is equipped with an infrared detector to search objects, a camera to take snapshots, and a stepper motor to support rotation. Each sensor is rotated periodically and during each period the sensor can monitor the objects that it takes care of for a portion of time. We call this monitoring behavior *the temporal coverage model*, whose goal is to guarantee each object to be monitored by sensors for at least a threshold ratio of time per period. The temporal coverage model has practical applications in, for example, monitoring the sea surface or underwater objects by sonar sensors [7], [8] and detecting weather changes or ground objects by airborne radars [9], [10].

From the above discussion, we define an *R&D sensor deployment problem*, which tries to deploy the minimum number of sensors to cover a given set of objects to satisfy their temporal coverage requirements. In particular, the sensing coverage of each sensor is modeled by a *sector* with an opening angle θ. Each sensor has 360 degrees of freedom to rotate and a sensing range of r_s, as shown in Fig. 1a. The time axis is divided into fixed-length *frames* and in each frame a sensor rotates one round and spends total (constant) time T in monitoring objects (here we ignore the rotation time for simplicity). Fig. 1b gives two examples, where sensor s_i stops two times per frame to monitor the objects in sectors A and B and sensor s_j stops four times per frame to monitor the objects in sectors C, D, E, and F. Fig. 1c shows the temporal coverage provided by s_i and s_j if they evenly divide their monitoring time to sectors. An object is said to be *δ-time covered*, $0 < \delta \leq 1$, if during each frame it is monitored by sensors for at least δT time. An R&D sensor network is said to achieve *δ-time coverage* if all objects are δ-time covered in every frame. Then, our R&D sensor deployment problem asks how to calculate the

- *Y.-C. Wang is with the Department of Computer Science and Engineering, National Sun Yat-Sen University, No. 70, Lienhai Road, Kaohsiung 80424, Taiwan, R.O.C. E-mail: ycwang@cse.nsysu.edu.tw.*
- *Y.-F. Chen and Y.-C. Tseng are with the Department of Computer Science, National Chiao-Tung University, 1001 University Road, Hsinchu 30010, Taiwan, R.O.C. E-mail: {yungfu, yctseng}@cs.nctu.edu.tw.*

Manuscript received 21 July 2010; revised 3 July 2011; accepted 15 July 2011; published online 1 Aug. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-07-0348. Digital Object Identifier no. 10.1109/TMC.2011.161.

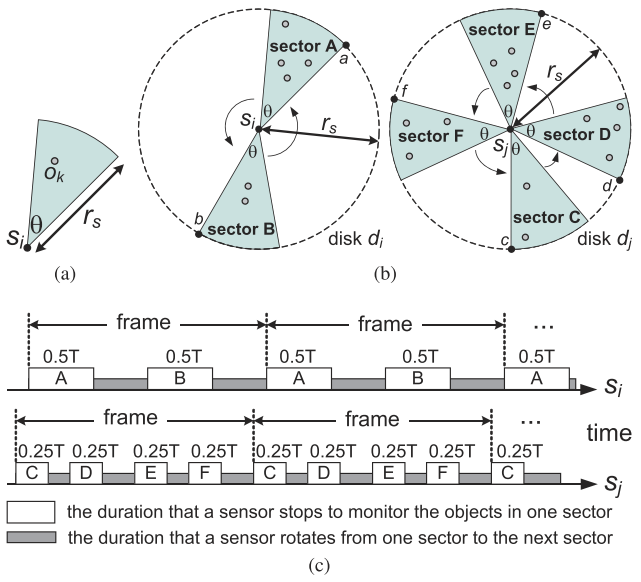


Fig. 1. The sensing model of R&D sensors. (a) Sensing coverage. (b) Sector coverage. (c) Temporal coverage.

minimum number of R&D sensors and determine their locations and rotation schedules to achieve δ-time coverage of the network. This problem is NP-hard because we can reduce the *geometric disk cover (GDC) problem* [11], which is NP-complete, to one of its instances.

To solve the R&D sensor deployment problem, we propose two heuristics. The idea is to use GDC to calculate disks to cover all objects and then select a *subset* of disks to deploy with R&D sensors such that all objects are δ-time covered. Specifically, the *maximum covering deployment (MCD) heuristic* suggests deploying sensors on the disks that cover more objects. In this way, when objects congregate together, MCD can deploy fewer sensors to satisfy their δ-time coverage requirements. On the other hand, the *disk-overlapping deployment (DOD) heuristic* prefers deploying sensors to cover the joint sectors of overlapped disks. Therefore, when objects are arbitrarily placed in the sensing field, DOD can take advantage of disk overlap to reduce the number of sensors. To summarize, the paper makes two major contributions. First, we define a new temporal coverage model, which could have many practical applications in visual monitoring, sonar, and radar systems. An even-driven surveillance prototype is also developed to demonstrate its feasibility. Second, we formulate a new R&D sensor deployment problem and propose two efficient heuristics to save the deployment cost. Both analytical and simulation results are presented to verify their effectiveness.

We organize the paper as follows: literature survey is conducted in Section 2. Section 3 gives the implementation of our event-driven surveillance prototype and defines the R&D sensor deployment problem. Section 4 proposes the MCD and DOD heuristics. Performance evaluations are presented in Section 5. Section 6 concludes the paper and gives future research topics.

2 Related Work

The subjects of sensing coverage and network deployment in WSNs have been widely studied. Below, we first discuss the sensing coverage issues, and then review the deployment schemes in both omnidirectional and directional WSNs.

2.1 Sensing Coverage Issues

Conventional research on sensing coverage usually focuses on *full coverage* in regular regions such as a rectangle. Most studies aim at the *k-coverage problem* whose goal is to activate a subset of sensors to make every location in the sensing field be monitored by at least k sensors. Solutions for omnidirectional sensors [12], [13] and directional sensors [14] have been developed. In addition, Kumar et al. [15] propose *barrier coverage* for long-thin belt regions, where a belt region is said to be *k-barrier covered* by an omnidirectional WSN if all crossing paths through the region are k-covered. Since barrier coverage focuses on crossing paths in belt regions, existing solutions of the k-coverage problem may not be directly applied. The result of barrier coverage is also extended to directional sensors [16].

Given an omnidirectional WSN in a rectangular region, [17], [18] consider the problem of finding the *minimal/maximal exposure paths*. Specifically, when an intruder selects any point to enter the region and crosses it, the minimal/maximal exposure path is a path that the intruder is monitored by the minimum/maximum number of sensors, which can be viewed as the worst case/best case coverage of the network. How to find minimal exposure paths in directional WSNs is discussed in [19]. Gui and Mohapatra [20] propose an intruder tracking problem whose goal is to find the minimum number of omnidirectional sensors such that if they are uniformly distributed, the average length of an uncovered path traveled by the intruder is always shorter than a threshold. In addition, Liu et al. [21] calculate the minimum node density required by a directional WSN to make all moving objects be detected by sensors.

Several studies address *time-related* coverage issues in omnidirectional WSNs. Gui and Mohapatra [22] consider that at any time instance, only a very small subset of sensors are activated to extend the network lifetime, where they form an *active zone* to monitor the sensing field. As the time progresses, the active zone moves along a predefined path. Assuming that sensors arbitrarily move in a region, Liu et al. [23] investigate the network coverage problems caused by the continuous movement of sensors such as the fraction of covered area and the time used to detect moving objects. Chang et al. [24] consider a mobile WSN with a big coverage hole and the number of sensors is not enough to cover the hole. Then, the hole "migrates" by the movement of sensors such that every location is covered by sensors for at least a portion of time. Liu and Cao [25] define a *spatial-temporal coverage metric*, where the metric of a small area is the product of the area size and the period during which the area is covered. Then, the goal is to activate a subset of sensors to maximize the overall metric of all areas. However, none of the above work considers time-related coverage issues in directional WSNs. Therefore, this motivates us to investigate the temporal coverage issue by exploiting the rotation of R&D sensors.

2.2 Deployment Schemes in Omnidirectional WSNs

The deployment problem in omnidirectional WSNs has been studied in a variety of contexts. Several studies model

a region by grid points and deploy sensors on a subset of grid points. Chakrabarty et al. [26] consider two types of sensors with different costs and sensing ranges, and the goal is to make each grid point be k-covered with the minimum cost. Considering a probabilistic sensing model, Dhillon and Chakrabarty [27] deploy sensors to make each grid point be covered by sensors with the minimum confidence level. Lin and Chiu [28] suggest deploying sensors to make every grid point be covered by different sensors, so the result is used to distinguish from different grid points for localization applications. Without the help of grid points, [29], [30] propose an optimal deployment scheme to provide 1-coverage of the sensing field, where sensors are deployed in a strip-by-strip manner. In [31], a hexagon-like deployment is proposed to provide k-coverage of a region. A framework to evaluate network lifetimes and costs for various deployments is proposed in [32].

Mobilizers are assumed in many studies to automate the deployment. When events occur, Butler and Rus [33] move a subset of sensors to detect the events while maintains 1-coverage of the sensing field. Imaging that sensors exert attractive or repulsive forces on each other, Zou and Chakrabarty [34] exploit such *virtual forces* to help deploy a WSN. Virtual forces are also adopted in [35] and [36]. In addition, grid structures [37] and Voronoi diagrams [38] are adopted to facilitate the deployment procedure, where sensors are moved from the high-density grids or small cells to other places so that they can be evenly deployed in the sensing field. Wang et al. [39] calculate the minimum number of locations to deploy with sensors to provide 1-coverage of the sensing field, and then moves sensors to these locations such that they can consume the least amount of energy. Designing in a similar way, Wang and Tseng [40] extend the result to provide k-coverage of the sensing field.

2.3 Deployment Schemes in Directional WSNs

Given a set of static targets, Ai and Abouzeid [41] consider a *maximum coverage with minimum sensors problem* whose goal is to activate the minimum number of directional sensors to cover the maximum number of targets. This problem is formulated by integer linear programming and a greedy strategy by selecting sensors to cover more objects is proposed. Assuming that directional sensors can switch to specific directions, Cai et al. [42] organize the directions of sensors into multiple *cover sets*, where in each cover set all targets are covered by the corresponding sensors. Then, a cover set is selected (and the corresponding sensors are activated) in each period to extend the network lifetime. Ma and Liu [43] consider that sensors have directional *sensing* and *communication* ranges and model the WSN by a directed communication graph to check the connectivity among sensors. Assuming that directional sensors can only rotate within a constrained angle and the network has a subset of anchor sensors with known positions, Lee [6] considers a localization problem to calibrate the positions of nonanchor sensors. Each sensor measures the relative ranges from other sensors and compensates the confined field of view by rotating. However, these studies do not consider the R&D sensor deployment problem.

How to deploy *static* directional sensors is also discussed in the literature. Han et al. [44] solve both the *connected point-coverage* and *connected region-coverage* problems, which seek to deploy the minimum number of directional sensors to construct a connect network that covers a given set of point-locations and an infinite 2D plane, respectively. For the connected point-coverage problem, the work first finds the sectors anchored by one, two, and three point-locations that cover the maximum number of point-locations and then deploys directional sensors to cover these sectors. Then, extra sensors are added to maintain the network connectivity. For the connected region-coverage problem, sectors are placed in a hexagon-like fashion to cover the 2D plane. Osais et al. [45] model the sensing field by a set of points on which directional sensors can be deployed. Given a subset of critical points, the work deploys the minimum number of sensors to cover all critical points by integer linear programming. However, once sensors are deployed, they do not rotate in [44] and [45]. Djugash et al. [46] install multiple directional sensors of the same type on one node, where each sensor faces to a different direction. In this way, one may adopt any omnidirectional WSN deployment solution to cover all objects. However, because a sensor is usually more expensive than a stepper motor, this solution may encounter a higher hardware cost compared with that using R&D sensors. In addition, under the temporal coverage model, since objects need not be always covered by sensors, we can trade the rotation time for reducing the hardware cost. Therefore, in this paper, we propose R&D sensor deployment heuristics to realize the temporal coverage model and reduce the deployment cost.

3 An R&D Sensor Surveillance Prototype and Its Deployment Problem

Below, to demonstrate how R&D sensors can be applied to indoor applications, we first develop a prototype of sensors and objects and integrate them to organize an even-driven surveillance system. We then formally define our R&D sensor deployment problem.

3.1 Prototyping an R&D Sensor Surveillance System

Conventional surveillance systems typically collect a large amount of videos from wall-mounted cameras, which requires huge computation to analyze [47]. To reduce the video content produced, we develop an event-driven surveillance system by R&D sensors, whose architecture is shown in Fig. 2. Specifically, we consider an indoor environment containing static sensors and objects to be monitored. Each sensor is equipped with an infrared detector, a camera, a wireless interface, and a stepper motor. The infrared detector has a directional sensing range and is used to search and detect objects. The camera has a zooming capability to take different resolutions of snapshots. The wireless interface is used to interact with the server and transmit messages. The stepper motor supports the rotation of a sensor. We use a security scenario to illustrate how this system works (refer to Fig. 2). In particular, each sensor behaves as follows:

- Initially, each sensor rotates one round to scan all objects around it. Using the north direction as the

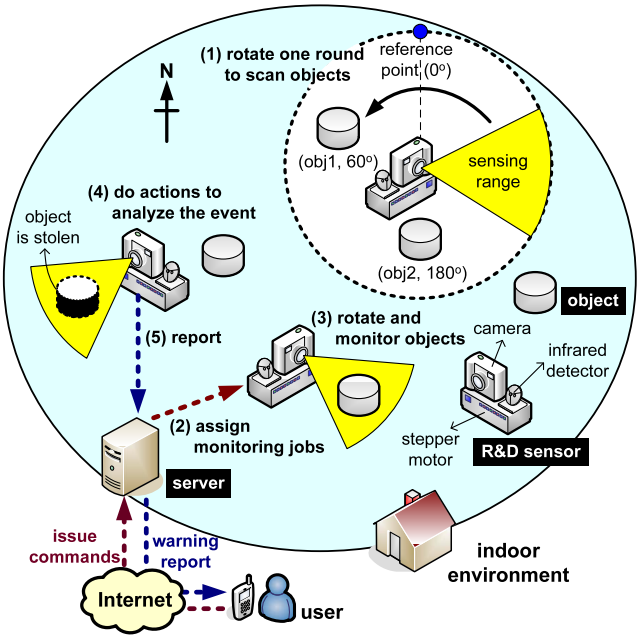


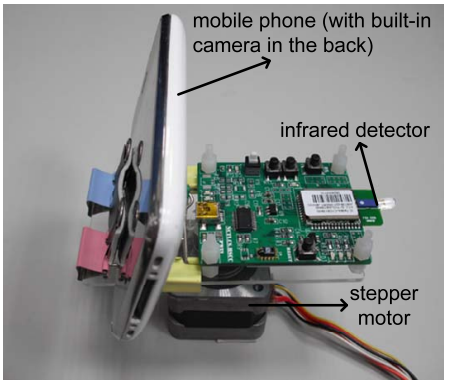
Fig. 2. The event-driven surveillance system by R&D sensors.

reference, the sensor records its surrounding objects along with their angles and then reports these information to the server.

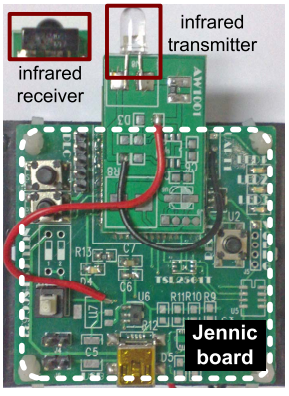
- After collecting the data from all sensors, the server assigns the objects to be monitored by each sensor and broadcasts its assignments. Then, sensors start to monitor objects according to their schedules. During rotation, each sensor can command its camera to zoom in/out and take snapshots according to the application scenario.
- When an object is stolen, the responsible sensor can soon learn the fact due to the loss of infrared signal. Then, many actions could be taken. For example, the sensor can notify the neighboring sensors of turning their cameras toward the original position of the stolen object to analyze the reason of this event.

We prototype R&D sensors and objects to realize the above scenario. Each R&D sensor is composed of an infrared detector, a stepper motor, and a camera, as shown in Fig. 3a. The infrared detector (shown in Fig. 3b) consists of an infrared transmitter and a Jennic board [48]. The infrared transmitter has a beam angle of 15 degrees and emits an infrared signal every 50 milliseconds to search the surrounding objects. The Jennic board supports the ZigBee short-range communication [49] to exchange messages with other sensors. We adopt the SANYO stepper motor (shown in Fig. 3c) to support the rotation of sensors. It turns 1.8 degrees per step and is controlled by a stepper motor driver. Each driver manages two stepper motors simultaneously. We adopt a mobile phone with built-in camera to take snapshots, so it can transmit the snapshots to the server through its 3G interface.

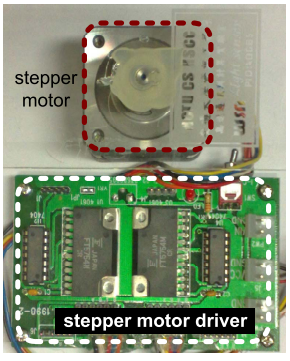
On the other hand, in order to detect objects, each object is attached with an *object module*. This module consists of an infrared receiver (shown in Fig. 3b) and a Jennic board. When an R&D sensor rotates to monitor an object, since the corresponding object module will receive the infrared signal emitted from the sensor, this module can notify the



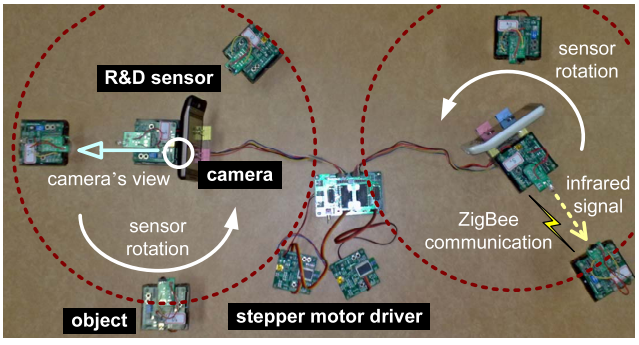
(a)



(b)



(c)



(d)

Fig. 3. Our implementation of the event-driven surveillance system. (a) The R&D sensor. (b) The infrared detector and the object module. (c) The stepper motor (with the driver). (d) A bird's-eye view of our prototype.

sensor by sending a message through its Jennic board. Here, we adopt the 940 nm wavelength module as the infrared receiver, whose sensing angle is 45 degrees and maximum receipt distance is 10 meters. When an object is taken away, the sensor cannot hear the feedback from the object module through the Jennic board. Thus, the sensor can know that the object disappears. Fig. 3d shows a bird's-eye view of our prototype, where a dashed circle is the sensing range of an R&D sensor when it rotates one round.

3.2 R&D Sensor Deployment Problem

We are given a set of static objects $\widehat{\mathcal{O}} = \{o_1, o_2, \ldots, o_m\}$ modeled by point-locations in a 2D plane. Each sensor has a sensing range modeled by one *sector* with an opening angle of $\theta \in (0, \pi)$ and a radius of r_s, and an omnidirectional

communication range whose radius is r_c. To formulate the problem, we make the following assumptions:

- All sensors have the same θ, r_s, and r_c values, but the relationship between r_s and r_c is arbitrary. We consider the binary sensing model,[1] where an object is said to be *covered* by a sensor if it locates in the sensing coverage of the sensor. Fig. 1a gives an example, where object o_k is covered by sensor s_i. We assume that the monitoring results of sensors is analyzed by a powerful server. Therefore, each sensor takes the equal time to detect objects, no matter the number of objects to be monitored. For example, camera or sonar sensors satisfy this assumption since the server can analyze the objects by image processing or sonar recognition.
- All sensors have the same rotation speed and 360 degrees of freedom to rotate *counterclockwise*.[2] Also, each sensor is allowed to stop during rotation. The time for a sensor to rotate one round is called a *frame*. Although the rotation angle between any two adjacent stopping points is different, a sensor must rotate 360 degrees in each frame. Fig. 1b gives an example. Sensor s_i has two stopping points a and b, so it takes $0.5T$ time to monitor the objects in sector A, rotates by $\angle as_ib$, stops to monitor the objects in sector B for $0.5T$ time, and then rotates by $\angle bs_ia$. Thus, s_i totally rotates by $\angle as_ib + \angle bs_ia = 360^\circ$ in each frame. Similarly, supposing that sensor s_j starts rotating at point c, it totally rotates by $\angle cs_jd + \angle ds_je + \angle es_jf + \angle fs_jc = 360^\circ$ in each frame. Since the rotation speed is the same, the total time spent on rotating 360 degrees (without stopping) in each frame must be the same. Note that the rotation angle to move from one sector to the next sector is no smaller than θ since sectors are not overlapped in a disk.
- Stepper motors have a correction mechanism. For example, each stepper motor can be equipped with a compass to correct the direction during rotation. Thus, we ignore the rotation error caused by stepper motors. In addition, stepper motors are powered by large-capacity batteries, so we do not take their energy consumption into account.
- Sensors are time-synchronized for detection purpose, because they need to attach time stamps on the monitoring data and forward them to the server. All sensors have the same frame length but their frames are not necessarily aligned. Besides, we assume that sensors can be precisely deployed on any location in the sensing field.

When a sensor s_i rotates one round (without changing its location), its sensing coverage scans a complete disk d_i that is centered at s_i and with a radius of r_s. According to the objects in d_i, we cut it into multiple *disjointed* sectors, where each sector has an opening angle of θ. Sensor s_i then rotates its sensing coverage to fit each sector. When its sensing coverage fits a sector, we say that s_i *covers* the sector. Fig. 1b gives two examples. Disk d_i is cut into two sectors and sensor s_i rotates to cover sectors A and B, while disk d_j is cut into four sectors and sensor s_j rotates to cover sectors C, D, E, and F.

We divide the time axis into fixed-length *frames*. In each frame, a sensor s_i rotates one round to cover the objects in its disk d_i. Supposing that d_i has α_i sectors, s_i should stop to cover each sector for $\frac{T}{\alpha_i}$ time and then rotate to the next sector. Therefore, the frame length is the total time that a sensor stops to cover all sectors (that is, T) and the time to rotate one round (marked as gray in Fig. 1c). Fig. 1c gives two examples. Since disks d_i and d_j are cut into two and four sectors, sensors s_i and s_j stop to cover each disk for $0.5T$ and $0.25T$ time, respectively. Given a threshold $0 < \delta \leq 1$, an object is said to be *δ-time covered* if and only if the object is covered by sensors for at least δT time in every frame. Fig. 1c gives two example, where the objects in disks d_i and d_j are 0.5-time and 0.25-time covered, respectively. Then, given $\widehat{\mathcal{O}}$ and δ, the *R&D sensor deployment problem* asks how to use the minimum number of R&D sensors and determine their locations and rotation schedules to cover all objects in $\widehat{\mathcal{O}}$ such that each object is δ-time covered.

Note that the rotational latency does not affect the time used to monitor objects, because each sensor has reserved time T to monitor all of its objects in every frame. In addition, since the frame length is fixed, each sensor is allowed to cover at most $\lfloor \frac{1}{\delta} \rfloor$ sectors in every frame. (Thus, a sensor stops at most $\lfloor \frac{1}{\delta} \rfloor$ times per frame.) When a disk is cut into more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, it requires multiple sensors to cover its sectors. Fig. 1b gives an example. Supposing that 0.5-time coverage is required (that is, $\delta = 0.5$), we need to deploy two sensors on disk d_j to make its objects be 0.5-time covered. Theorem 1 shows that the R&D sensor deployment problem is NP-hard.

Theorem 1. *The R&D sensor deployment problem is NP-hard.*

Proof. We reduce the GDC problem, which is NP-complete, to the R&D sensor deployment problem. Given a set of point-locations, GDC asks how to find the minimum number of disks to cover all point-locations. Here, we consider an instance of the R&D sensor deployment problem, where $\delta = \frac{\theta}{2\pi}$ and π is divisible by θ. We show that GDC has a solution if and only if the R&D sensor deployment problem has a solution.

Suppose that we have a solution to the R&D sensor deployment problem. Because $\delta = \frac{\theta}{2\pi}$, each sensor can rotate to cover $\frac{2\pi}{\theta}$ sectors in every frame. Since each sector has an opening angle of θ, these sectors form a complete disk. In other words, each sensor can cover all of the objects in its disk per frame. Therefore, these disks constitute a solution to GDC. This proves the *if* part.

Conversely, suppose that we have a solution to GDC, which is a set of disks. Then, by deploying a sensor on each disk in the set, we can guarantee that all objects are δ-time covered (since each sensor can rotate to cover all of the objects in its disk per frame). This constitutes a solution to the R&D sensor deployment problem, thus proving the *only if* part. □

1. When the sensing region is irregular, we can use a probabilistic model to describe it [39]. In this case, an object is said to be covered by a sensor if the detection probability of the sensor to the object exceeds a threshold.

2. This assumption is to guarantee that all frames have the same length. It may result in variable lengths of frames to allow sensors to rotate in arbitrary directions, and how to deal with the above issue is out of the paper's scope.

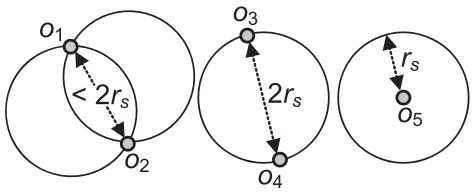


Fig. 4. An example of the modified GDC solution.

4 The Proposed Sensor Deployment Heuristics

Our idea is to first calculate disks to cover all objects and then select a subset of disks to deploy with R&D sensors such that all objects are δ-time covered. We propose two heuristics, where MCD suggests deploying sensors on the disks that cover more objects, while DOD exploits disk overlap by deploying sensors to cover joint sectors. Then, we add the minimum number of relay nodes to maintain the network connectivity.

4.1 Maximum Covering Deployment Heuristic

Given a set of objects $\widehat{\mathcal{O}}$, MCD has three phases.

- **Phase 1:** Find a set of disks $\widehat{\mathcal{D}}$ to cover all objects in $\widehat{\mathcal{O}}$.
- **Phase 2:** Set all objects in $\widehat{\mathcal{O}}$ to *unmarked*. We then iteratively select the disk with the maximum number of unmarked objects, conduct the *sector cutting operation* to find its sectors, and mark all of its objects. This iteration is repeated until all objects in $\widehat{\mathcal{O}}$ are marked. Then, we remove the disks without conducting the sector cutting operation from $\widehat{\mathcal{D}}$.
- **Phase 3:** Iteratively select the disk from $\widehat{\mathcal{D}}$ that covers the maximum number of objects. We then deploy one R&D sensor on the center of that disk and determine its rotation schedule. This iteration is repeated until all objects in $\widehat{\mathcal{O}}$ are covered.

Below, we discuss the detail of each phase. In phase 1, we modify the GDC solution in [50] to calculate a set of disks $\widehat{\mathcal{D}}$ to cover all objects in $\widehat{\mathcal{O}}$, which contains three rules.

1. If two objects have a distance smaller than $2r_s$, we place two disks such that their circumferences intersect at these two objects.
2. If two objects have a distance equal to $2r_s$, we place a disk such that its circumference passes these two objects.
3. If an object is *isolated*, in the sense that its distance to the nearest object is larger than $2r_s$, we place a disk such that its center locates at the object.

Fig. 4 gives three examples. Since we need to check each pair of objects in $\widehat{\mathcal{O}}$, the maximum number of disks in $\widehat{\mathcal{D}}$ is $2C_2^m$, where $C_2^m = \frac{m!}{2(m-2)!}$ is the combination of selecting two objects from m objects in $\widehat{\mathcal{O}}$.

Since the size of $\widehat{\mathcal{D}}$ is larger than m, which means that some disks are redundant, we select a subset of $O(m)$ disks from $\widehat{\mathcal{D}}$ and find their sectors in phase 2. Specifically, we iteratively select the disk that covers the maximum number of objects and then conduct the two-step sector cutting operation to find its sectors:

1. **Indexing and clustering.** We cluster objects according to their locations in the disk. Specifically, we arbitrarily select an object, index it by o_1, and add it to cluster 1. From o_1, we scan *nonindexed* objects counterclockwise. On finding a nonindexed object, we index it by o_2 and decide its cluster. If $\angle o_1 s_a o_2 \leq \theta$, o_2 is added to cluster 1; otherwise, o_2 is added to cluster 2, where s_a is the disk center. For a general case, supposing that we index an object by o_i and add it to cluster k, the next nonindexed object is indexed by o_{i+1}. If $\angle o_i s_a o_{i+1} \leq \theta$, o_{i+1} is added to cluster k; otherwise, o_{i+1} is added to cluster $k+1$. This operation is repeated until all objects are indexed. Figs. 5a and 5b together give an example by grouping all objects into three clusters.
2. **Deciding sectors.** We then decide the sectors to cover all objects in each cluster. Supposing that there are K clusters, we handle these clusters in the sequence of $K, 1, 2, \ldots, K-1$. For each cluster, starting from the *uncovered* object with the smallest index, say, o_i, we place a sector whose edge passes o_i such that the sector covers the maximum number of objects. This operation is repeated until all objects in the cluster are covered. Fig. 5c gives an example, where we place a sector A whose edge passes o_8 and covers all objects in cluster 3. Since sector A also covers o_1 in cluster 1, we place a sector B whose edge passes o_2 and covers the remaining objects in cluster 1.

Since we arbitrarily select the initial object in step 1, the included angle between clusters K and 1 may be no larger than θ. In this case, by selecting cluster K to start placing sectors, the last sector in cluster K could also cover some objects in cluster 1 (so the number of sectors is reduced). Note that we conduct the sector cutting operation on only $O(m)$ disks since $\widehat{\mathcal{O}}$ has m objects. Those disks without conducting the sector cutting operation are removed from $\widehat{\mathcal{D}}$,

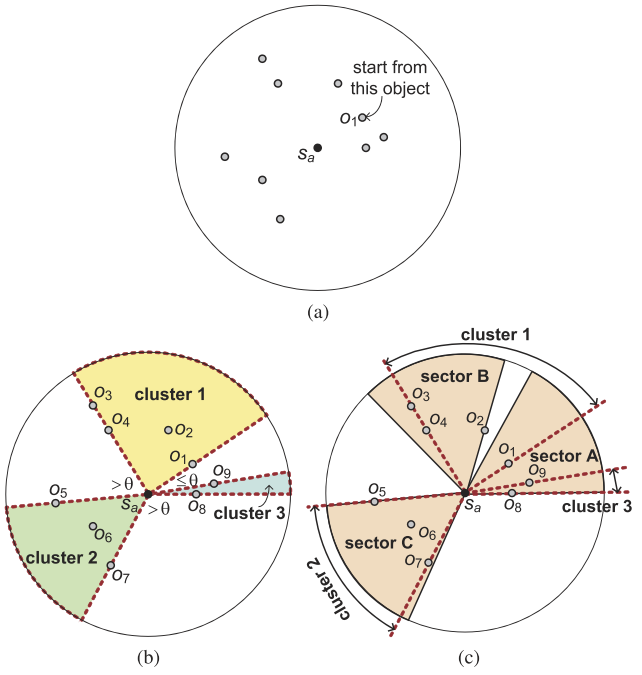


Fig. 5. The sector cutting operation on a disk. (a) Select an object and index it by o_1. (b) Scan all objects counterclockwise and cluster them accordingly. (c) Starting from the last cluster, place sectors to cover each cluster.

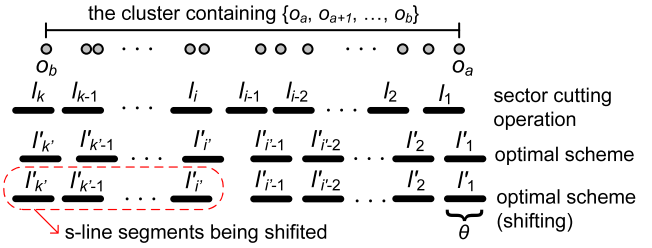


Fig. 6. Viewing a cluster as a line segment from objects o_a to o_b.

so the size of $\widehat{\mathcal{D}}$ can shrink from $O(2C_2^m)$ to $O(m)$. Theorem 2 shows that the sector cutting operation can place the minimum number of sectors to cover the objects in a disk.

Theorem 2. *The sector cutting operation places the minimum number of sectors to cover the objects in a disk.*

Proof. We divide the proof into two parts. First, we show that the sector cutting operation finds the minimum number of clusters in a disk, where 1) the included angle between any two adjacent clusters is larger than θ and 2) the included angle between any two adjacent objects in each cluster is smaller than or equal to θ. Second, for each cluster, the sector cutting operation places the minimum number of sectors to cover all of its objects.

For the first part, recall that the sector cutting operation starts placing sectors from the last cluster, so its last sector may also cover some objects in the first cluster. In this case, we can combine the first and last clusters into the same one. Without loss of generality, suppose that the sector cutting operation calculates a set of clusters $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, where the included angle between clusters c_K and c_1 is larger than θ. Then, suppose that there is a scheme which calculates a set of clusters $\mathcal{C}' = \{c'_1, c'_2, \ldots, c'_{K'}\}$, where $K' < K$ and the included angle between any two adjacent clusters is larger than θ (that is, Condition 1 is satisfied). In this case, there must be a cluster, say, c'_j in $\mathcal{C}'$ that overlaps at least two clusters c_i and c_{i+1} in $\mathcal{C}$. Let o_k and o_{k+1} be the last and first objects in clusters c_i and c_{i+1}, respectively. Both o_k and o_{k+1} must belong to c'_j. However, the included angle between o_k and o_{k+1} must be larger than θ (otherwise, they will be both included in c_i). Thus, this violates Condition 2 and causes a contradiction. Therefore, the sector cutting operation finds the minimum number of clusters in a disk.

For the second part, let us consider a cluster containing objects $\{o_a, o_{a+1}, \ldots, o_b\}$. For ease of presentation, we view the cluster as a line segment from o_a to o_b, as shown in Fig. 6. In this case, each sector used to cover objects can be viewed as a line segment whose length is θ. For convenience, we call such a line segment a *s-line segment*. Since any two adjacent objects in a cluster must have an included angle smaller than or equal to θ, the distance between any two adjacent s-line segments must be smaller than or equal to θ. Then, suppose that the sector cutting operation calculates a set of s-line segments $\mathcal{L} = \{l_1, l_2, \ldots, l_k\}$ and there is an optimal scheme which calculates another set of s-line segments $\mathcal{L}' = \{l'_1, l'_2, \ldots, l'_{k'}\}$. Each s-line segment $l'_{i'}$ in $\mathcal{L}'$ must overlap either exact one s-line segment l_i or two s-line segments l_i and l_{i+1} in $\mathcal{L}$. In the former case, $l'_{i'}$ is *equal* to l_i since we can shift $l'_{i'}$ such that its right-hand end aligns its rightmost object, as shown in Fig. 6. In addition, since the distance between any two adjacent s-line segments is smaller than or equal to θ, if $l'_{i'}$ is equal to l_i, then we can guarantee that $l'_{i'+1}$ is equal to $l_{i+1}, l'_{i'+2}$ is equal to $l_{i+2}, \ldots$, and $l'_{k'}$ is equal to l_k by doing shifting. Thus, we have $k' - i' + 1 = k - i + 1$. On the other hand, $l'_{i'-1}$ must overlap l_{i-1} and l_{i-2}, $l'_{i'-2}$ must overlap l_{i-2} and $l_{i-3}, \ldots$, and l'_2 must overlap l_2 and l_1. Since l'_2 cannot cover o_a (otherwise, it cannot overlap both l_2 and l_1), we need to use one extra s-line segment (that is, l'_1) to cover o_a. In this case, we have $i' - 1 = i - 1$ and thus $k = k'$. Therefore, the sector cutting operation places the minimum number of sectors to cover the objects in a cluster.

By proving both the two parts, we show that the sector cutting operation places the minimum number of sectors to cover the objects in a disk. □

Finally, in phase 3, we deploy R&D sensors on disk centers to cover objects. We sort all disks in $\widehat{\mathcal{D}}$ by their maximum numbers of objects covered by *one sensor* in a decreasing order. In particular, if a disk has more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, this value is the maximum number of objects covered by its $\lfloor \frac{1}{\delta} \rfloor$ sectors; otherwise, it is the total number of objects in the disk. Then, we iteratively pick one disk, deploy a sensor on its center, and make the sensor rotate to monitor each of its sectors with equal time. Note that if the disk has more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, the sensor only monitors the $\lfloor \frac{1}{\delta} \rfloor$ sectors with the maximum number of objects. Then, we remove the objects covered by the sensor from $\widehat{\mathcal{O}}$. If d_i contains no objects, we remove it from $\widehat{\mathcal{D}}$. The above iteration is repeated until $\widehat{\mathcal{O}}$ becomes empty. Fig. 1b gives an example, where $\delta = 0.5$. We deploy one sensor to cover sectors D and E, then one sensor to cover sectors A and B, and finally one sensor to cover sectors C and F. All sensors monitor each sector for $0.5T$ time. Theorem 3 calculates the time complexity of MCD.

Theorem 3. *MCD has time complexity of* $O(m^2)$.

Proof. In phase 1, running the modified GDC solution takes $O(C_2^m)$ time since we need to search any pair of objects from $\widehat{\mathcal{O}}$. In phase 2, we build a maximum binary heap to maintain the disks in $\widehat{\mathcal{D}}$ (whose size is $O(2C_2^m)$), which takes $O(2C_2^m)$ time. Since deleting the maximum from the heap spends $O(\lg 2C_2^m)$ time and we delete $O(m)$ disks from the heap, it takes $O(m) \cdot O(\lg 2C_2^m) = O(m \lg m)$ time to do all deletions. In addition, the sector cutting operation totally takes $O(2m)$ time since in each step we scan m objects in $\widehat{\mathcal{O}}$. In phase 3, a maximum binary heap is also built in $O(m)$ time to maintain the disks in $\widehat{\mathcal{D}}$ (whose size is shrunk to $O(m)$). Since $\widehat{\mathcal{D}}$ has $O(m)$ disks, the number of heap operations to delete the maximum and to insert entries does not exceed $O(m)$, where each heap operation takes $O(\lg m)$ time. So, it totally takes $O(m) \cdot O(\lg m) = O(m \lg m)$ time to select disks in phase 3. Therefore, MCD has time complexity of $O(C_2^m) + O(2C_2^m) + O(m \lg m) + O(2m) + O(m) + O(m \lg m) = O(m^2)$. □

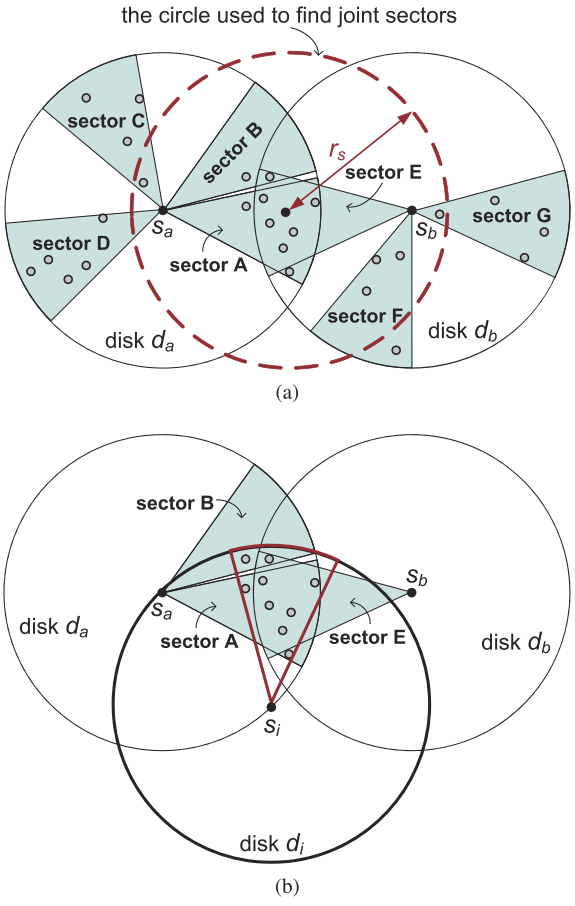


Fig. 7. An example of DOD. (a) Find the joint sectors of disks d_a and d_b. (b) Deploy one sensor at s_i to cover all objects in the joint sectors.

We remark that when objects congregate together, a sector could cover more objects. Thus, the number of sectors in each disk may be reduced. Therefore, by deploying sensors to cover the sectors with more objects, MCD can reduce the number of sensors.

4.2 Disk-Overlapping Deployment Heuristic

When objects are arbitrarily placed in the sensing field, each sector may cover only few objects. In this case, each disk requires more sectors to cover its objects and thus we may need to deploy multiple sensors on each of most disks. Therefore, the efficiency of MCD may degrade. To handle this case, we propose DOD whose idea is to exploit disk overlap to save sensors. Fig. 7 gives an example, where $\delta = 0.5$. Both disks d_a and d_b are divided into more than two sectors but a sensor can cover only two sectors. Thus, MCD deploys totally four sensors at s_a and s_b to cover all objects. However, since sectors A, B, and E overlap with each other, we can use one sensor to cover their objects. Specifically, we deploy one sensor at s_i to cover the objects in sectors A, B, and E, one sensor at s_a to cover sectors C and D, and one sensor at s_b to cover sectors F and G, which totally requires only three sensors. Here, sectors A, B, and E are called *joint sectors* since they can be "jointly" covered by one disk.

Based on the above observation, DOD exploits disk overlap by deploying sensors to cover joint sectors, which is outlined as follows:

- **Phase 1:** Use the modified GDC solution to calculate the set $\widehat{\mathcal{D}}$. We then select a subset of disks $\widehat{\mathcal{D}}_s$ from $\widehat{\mathcal{D}}$ to cover all objects in $\widehat{\mathcal{O}}$.
- **Phase 2:** Find the joint sectors of any two disks in $\widehat{\mathcal{D}}_s$.
- **Phase 3:** Calculate the *additional* disks used to cover the joint sectors. We then deploy R&D sensors on these additional disks and a subset of disks in $\widehat{\mathcal{D}}_s$.

Below, we discuss the detail of each phase. In phase 1, recall that the modified GDC solution finds a set $\widehat{\mathcal{D}}$ with $O(2C_2^m)$ disks. Then, from $\widehat{\mathcal{D}}$, we find a subset of disks $\widehat{\mathcal{D}}_s$ to cover all objects in $\widehat{\mathcal{O}}$ such that 1) the size of $\widehat{\mathcal{D}}_s$ is minimized and 2) the number of disks with no more than $\lfloor \frac{1}{\delta} \rfloor$ sectors is maximized. Here, we want to use the minimum number of disks to cover all objects by objective 1. On the other hand, when a disk has no more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, we can deploy one sensor to cover all of its sectors. So, objective 2 is to select as many such disks as possible. Therefore, we propose a *disk selection scheme* to achieve the two objectives.

1. Let $\widehat{\mathcal{D}}_s = \emptyset$ and set all objects in $\widehat{\mathcal{O}}$ to *unmarked*.
2. Sort the disks in $\widehat{\mathcal{D}}$ by their numbers of unmarked objects in a decreasing order. Then, we select the first $n > 1$ disks from $\widehat{\mathcal{D}}$ and conduct the sector cutting operation on each of these n disks *independently*.[3] To satisfy objective 2, two cases are considered.
   a. If some disks have no more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, we remove the disks with more than $\lfloor \frac{1}{\delta} \rfloor$ sectors from these n disks.
   b. Otherwise, all disks must have more than $\lfloor \frac{1}{\delta} \rfloor$ sectors. In this case, no disks are removed.

   Among the remaining disks, we select the disk, say, d_i that covers the maximum number of unmarked objects. We then add d_i to $\widehat{\mathcal{D}}_s$ and mark all objects in d_i.
3. Repeat step 2 until all objects in $\widehat{\mathcal{O}}$ are marked.

Note that $\widehat{\mathcal{D}}_s$ has $O(m)$ disks since there are m objects in $\widehat{\mathcal{O}}$. Then, in phase 2, we find the joint sectors of any two disks in $\widehat{\mathcal{D}}_s$ by Definition 1.

Definition 1. *Two disks have joint sectors if and only if 1) the distance between their centers is no larger than $2r_s$, 2) both disks have more than $\lfloor \frac{1}{\delta} \rfloor$ sectors, and 3) when drawing a circle between the two disks with a radius of r_s, each disk has at least one sector whose objects all locate inside the circle.*

Here, Condition 1 means that both disks overlap with each other. Condition 2 means that both disks require multiple sensors to cover their sectors, so it is efficient to deploy sensor(s) to *jointly* cover their sectors. Condition 3 indicates how to find joint sectors. Fig. 7a shows an example, where $\delta = 0.5$. Given two overlapped disks d_a and d_b (so Condition 1 is satisfied), both disks have more than $\lfloor \frac{1}{\delta} \rfloor = 2$ sectors (so Condition 2 is satisfied). Then, we draw a circle between d_a and d_b with a radius of r_s. In this case, all of the objects in sectors A, B, and E locate inside the circle. By Condition 3, they are joint sectors. All other sectors are called *nonjoint sectors*. For example, sectors C, D,

3. Unlike MCD, after conducting the sector cutting operation, we do not mark the objects in the disk. Thus, if an object locates in two disks, it is considered by both disks when conducting the sector cutting operation.

F, and G are nonjoint sectors because they have objects outside the circle.

Finally, in phase 3, we find the additional disks to cover the joint sectors and select disks to deploy with sensors. This phase involves three steps.

1. For each disk d_i in $\widehat{\mathcal{D}}_s$ *without* joint sectors, we deploy $\lceil \alpha_i \delta \rceil$ sensors to cover all of its sectors, where α_i is the number of nonjoint sectors in d_i. We then remove all objects in d_i from $\widehat{\mathcal{O}}$ and d_i from $\widehat{\mathcal{D}}_s$.
2. After step 1, $\widehat{\mathcal{D}}_s$ must remain only the disks with joint sectors. We then deploy sensors to cover the *nonjoint sectors* of these disks. For each disk in $\widehat{\mathcal{D}}_s$ with α_i nonjoint sectors, we deploy $\lceil \alpha_i \delta \rceil$ sensors to cover all of its nonjoint sectors. Here, since each sensor can cover $\lfloor \frac{1}{\delta} \rfloor$ sectors, there could be one sensor that covers only, say, $\gamma < \lfloor \frac{1}{\delta} \rfloor$ nonjoint sectors. In this case, we make this sensor *also* cover the $(\lfloor \frac{1}{\delta} \rfloor - \gamma)$ joint sectors with the maximum number of objects in that disk. Then, we remove all objects covered by these sensors from $\widehat{\mathcal{O}}$. Thus, the sectors and disks without objects are also removed accordingly. Fig. 7a gives an example, where $\delta = 0.5$. We deploy one sensor at s_a to cover the nonjoint sectors C and D of disk d_a and one sensor at s_b to cover the nonjoint sectors F and G of disk d_b. Then, only joint sectors A, B, E are left, as shown in Fig. 7b.
3. After step 2, each disk in $\widehat{\mathcal{D}}_s$ must remain only joint sectors. Thus, we iteratively select the two overlapped disks in $\widehat{\mathcal{D}}_s$ such that their joint sectors have the maximum number of objects. Let $\widehat{\mathcal{O}}_c$ denote the set of objects in these joint sectors. We then find a set of *additional* disks to cover the objects in $\widehat{\mathcal{O}}_c$. Here, recall that we have already calculated the set of disks $\widehat{\mathcal{D}}$ to cover all objects in $\widehat{\mathcal{O}}$ by the modified GDC solution in phase 1. So, we need not recalculate these additional disks since they must belong to $\widehat{\mathcal{D}} - \widehat{\mathcal{D}}_s$. Then, among these additional disks, we select the disk, say, d_i that covers all objects in $\widehat{\mathcal{O}}_c$ by using the minimum number of sectors (this can be calculated by the sector cutting operation). We then deploy sensor(s) to cover all sectors of d_i and remove $\widehat{\mathcal{O}}_c$ from $\widehat{\mathcal{O}}$. In this case, the sectors and disks without objects are also removed accordingly. Fig. 7b gives an example, where we deploy one sensor on the additional disk d_i to cover all objects in sectors A, B, and E. The above iteration is repeated until $\widehat{\mathcal{O}}$ is empty.

Note that in step 3, we do not simply deploy sensor(s) *between* the two overlapped disks since it may waste sensors. Fig. 7a gives an example, where $\delta = 0.5$. If we simply deploy sensor(s) at the center of the dash circle, we need to cut the circle into three sectors to cover all objects in the joint sectors A, B, and E. In this case, we have to deploy two sensors at the circle center. Instead, we just deploy one sensor at s_i to cover all objects in these joint sectors, as shown in Fig. 7b. Theorem 4 calculates the time complexity of DOD.

Theorem 4. *DOD has time complexity of* $O(m^2 + nm \lg m)$, *where* $n < m$ *is a system constant.*

Proof. In phase 1, running the modified GDC solution takes $O(C_2^m)$ time. To calculate $\widehat{\mathcal{D}}_s$, we build a maximum binary heap to maintain the disks in $\widehat{\mathcal{D}}$, which takes $O(2C_2^m)$ time. It involves n times of deleting-maximum operations and $n-1$ times of insertion operations to add one disk in $\widehat{\mathcal{D}}_s$, where each heap operation spends $O(\lg 2C_2^m) = O(\lg m)$ time. Since $\widehat{\mathcal{O}}$ has m objects, we add $O(m)$ disks to $\widehat{\mathcal{D}}_s$. Thus, calculating $\widehat{\mathcal{D}}_s$ totally spends $O(2C_2^m) + O(m) \cdot (O(n \lg m) + O((n-1) \lg m)) = O(m^2 + nm \lg m)$ time. Then, finding the joint sectors of any two disks in phase 2 takes $O(C_2^m)$ time since $\widehat{\mathcal{D}}_s$ has $O(m)$ disks. Finally, in phase 3, it takes $O(m)$ time to deploy sensors to cover the disks without joint sectors in step 1. Similarly, it takes $O(m)$ time to deploy sensors to cover the nonjoint sectors in step 2. In step 3, finding the additional disks to cover joint sectors takes no time since they belong to $\widehat{\mathcal{D}} - \widehat{\mathcal{D}}_s$. In addition, running the sector cutting operation on the selected disk requires $O(m)$ time. Since objects are removed whenever we deploy sensors to cover them, step 3 runs $O(m)$ iterations. So, the total time of step 3 is $O(m) \cdot O(m) = O(m^2)$. Therefore, the time complexity of DOD is $O(C_2^m) + O(m^2 + nm \lg m) + O(C_2^m) + O(m) + O(m) + O(m^2) = O(m^2 + nm \lg m)$. □

We remark on the difference between MCD and DOD. First, they take different strategies to select a subset of disks from $\widehat{\mathcal{D}}$ to deploy with sensors. MCD suggests selecting the disks that cover more objects. DOD prefers selecting the disks with no more than $\lfloor \frac{1}{\delta} \rfloor$ sectors so that each disk can be deployed with just one sensor. Second, while MCD always deploys sensors to cover the sectors with the maximum number of objects, DOD tries to deploy sensors to cover the joint sectors of overlapped disks. Due to the different designs, MCD saves more sensors when objects congregate together, while DOD works better when objects are arbitrarily placed in the sensing field. The simulation results in Section 5 also support the remark.

4.3 Maintaining the Network Connectivity

Till now, our deployment heuristics focus on covering all objects, but the network may not be connected. To solve this problem, we add extra *relay nodes* to maintain the network connectivity, where a relay node is the communication module of a sensor.[4] This solution has two advantages. First, the deployment cost is reduced. Second, our heuristics can allow an arbitrary relationship between r_s and r_c.

We modify the approach in [44] to add relay nodes. Given a set of sensors $\mathcal{S}$ calculated by MCD or DOD, we construct the minimum spanning tree on $\mathcal{S}$. Then, for each tree edge whose length, say, l is larger than r_c, we deploy $(\lceil \frac{l}{r_c} \rceil - 1)$ relay nodes along that edge. The distance between any two adjacent relay nodes is r_c. This operation is repeated until all tree edges are checked. Since $\widehat{\mathcal{O}}$ has m objects, both MCD and DOD deploy no more than $O(m)$ sensors. By using the Prim's algorithm to calculate the minimum spanning tree, this approach has time complexity of $O(m^2)$.

4. In several sensor platforms such as Jennic [48], each sensor consists of *separable* sensing and communication modules.

5 Performance Evaluation

We evaluate the performances of our deployment heuristics in terms of the number of sensors by developing a simulator in C++. A sensing field with 400×400 rectangle is considered, in which there are static objects needed to be monitored by sensors. We use two distributions, namely *random* and *congregating* distributions, to model the placement of objects in the sensing field. Under the random distribution, objects are uniformly and arbitrarily placed in the sensing field. Under the congregating distribution, 10 positions are selected from the sensing field and then objects are arbitrarily placed around these positions. For each sensor, we set $r_s = 10$ and $r_c = 20$. In DOD, we set $n = 5$. The number of objects (m), the sector angle θ, and the δ value are varied in the simulations.

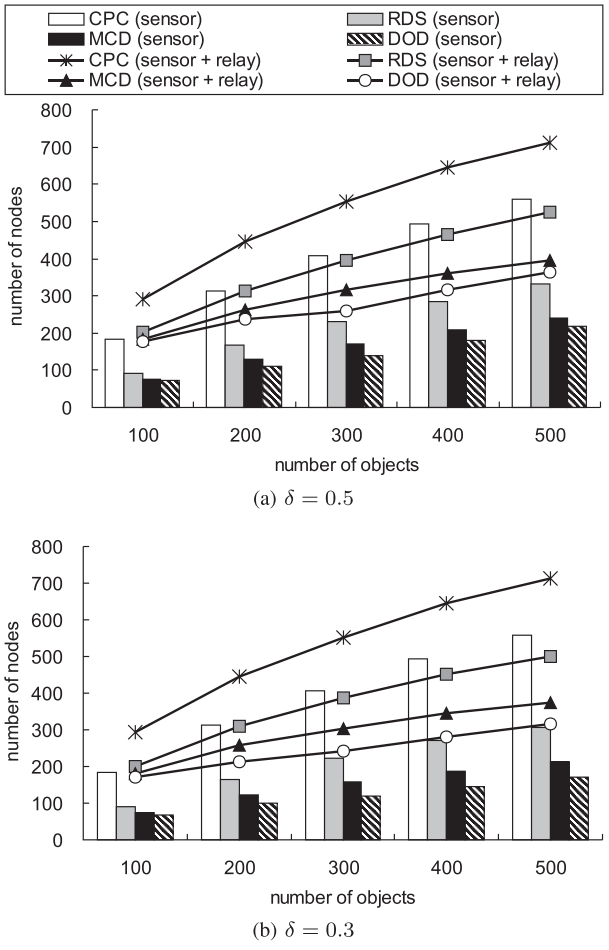
We compare MCD and DOD with the *connected point-coverage (CPC)* solution in [44], which seeks to deploy the minimum number of directional sensors to construct a connected network that covers all objects. However, once sensors are deployed, they do not rotate in CPC. According to the discussion in Section 2.3, since none of the existing work considers employing the rotation of R&D sensors to deploy a network, we thus develop a *random disk selection (RDS)* scheme for comparison. In RDS, a subset of objects are arbitrarily selected as disk centers such that these disks can cover all objects. Then, we find the sectors of each disk and deploy enough sensors to cover all of its sectors. Finally, we adopt the scheme in Section 4.3 to deploy relay nodes to maintain the network connectivity.

5.1 Effect of Different Object Numbers

We first investigate the effect of different object numbers on the number of sensors and relay nodes deployed by CPC, RDS, MCD, and DOD. The object number is ranged from 100 to 500. The δ value is set to 0.5 and 0.3 so that a sensor can rotate to cover two and three sectors, respectively. We set $\theta = 45°$ and measure 1) the number of sensors used to cover objects and 2) the total number of sensors and relay nodes used to construct the network.

Fig. 8 shows the number of nodes deployed by CPC, RDS, MCD, and DOD with different object numbers under the random distribution of objects. Explicitly, each scheme requires more sensors when the object number grows. In this case, we may need more relay nodes to connect these sensors. Since objects are arbitrarily placed and CPC does not allow sensors to rotate, it spends much more nodes compared with other schemes. The situation becomes worse when $\delta = 0.3$. In this case, RDS, MCD, and DOD allow a sensor to rotate to cover three sectors, while CPC limits a sensor to cover only one sector. This comparison shows the advantage of using R&D sensors in terms of the deployment cost.

In RDS, MCD, and DOD, a smaller δ value helps save more sensors because each sensor can rotate to cover more sectors. By carefully selecting the disks to deploy with sensors, MCD and DOD significantly outperform RDS. In addition, DOD uses fewer sensors compared with MCD due to two reasons. First, since objects are arbitrarily placed, each disk may cover only few objects. So, the greedy strategy of MCD that deploys sensors to cover the disks with more objects may not have significant effect. Second,



(a) $\delta = 0.5$

(b) $\delta = 0.3$

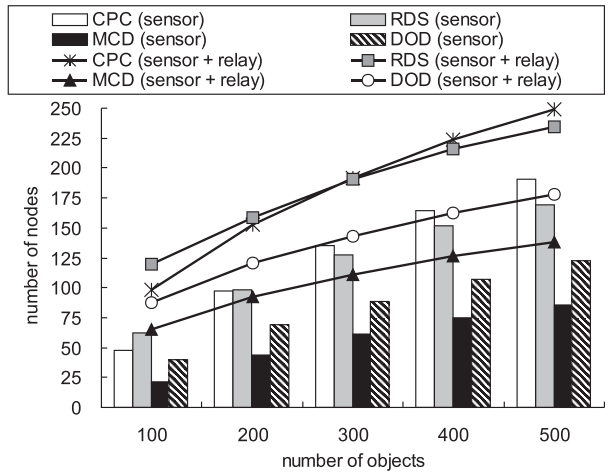
Fig. 8. The number of nodes deployed by CPC, RDS, MCD, and DOD with different object numbers under the random distribution of objects.

DOD exploits disk overlap by deploying sensors to cover joint sectors, which helps reduce the number of sensors.

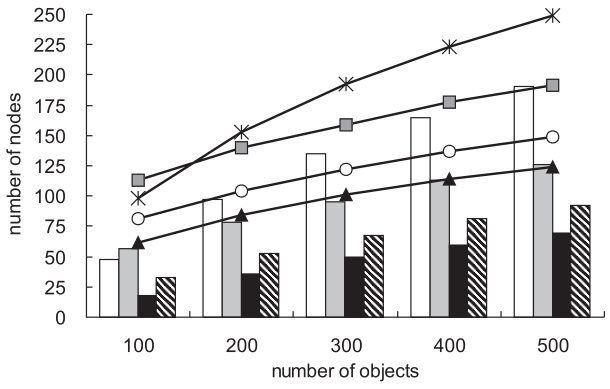
In average, when $\delta = 0.5$, MCD saves 58.1/24.2 percent of sensors and 42.0/18.5 percent of all nodes compared with CPC/RDS. On the other hand, DOD saves 63.1/33.2 percent of sensors and 47.9/26.8 percent of all nodes compared with CPC/RDS. When $\delta = 0.3$, MCD saves 61.3/27.3 percent of sensors and 43.9/19.6 percent of all nodes compared with CPC/RDS. On the other hand, DOD saves 68.3/40.3 percent of sensors and 52.3/31.6 percent of all nodes compared with CPC/RDS.

Fig. 9 shows the number of nodes deployed by CPC, RDS, MCD, and DOD with different object numbers under the congregating distribution of objects. Similarly, each scheme needs more nodes when there are more objects. Interestingly, when there are only 100 objects, CPC outperforms RDS. The reason is that since objects congregate together, CPC calculates relatively few sectors to cover them. On the other hand, RDS finds disks in a random manner so that more disks may be used. However, when the object number grows, CPC requires more nodes compared with RDS because RDS allows sensors to rotate to cover multiple sectors.

Recall that both MCD and DOD find a subset of disks from $\widehat{\mathcal{D}}$ to deploy with sensors. When objects congregate together, MCD selects the disks with more objects so that it



(a) $\delta = 0.5$



(b) $\delta = 0.3$

Fig. 9. The number of nodes deployed by CPC, RDS, MCD, and DOD with different object numbers under the congregating distribution of objects.

can find a smaller subset of disks. On the other hand, DOD prefers selecting the disks with no more than $\lfloor \frac{1}{\delta} \rfloor$ sectors (and then finds additional disks to cover joint sectors), so it may calculate a relatively larger subset of disks. Therefore, MCD saves more sensors compared with DOD under the congregating distribution of objects.

In average, when $\delta = 0.5$, MCD saves 54.9/54.7 percent of sensors and 40.7/42.3 percent of all nodes compared with CPC/RDS. On the other hand, DOD saves 30.0/30.7 percent of sensors and 22.8/24.9 percent of all nodes compared with CPC/RDS. When $\delta = 0.3$, MCD saves 63.4/52.8 percent of sensors and 45.8/38.5 percent of all nodes compared with CPC/RDS. On the other hand, DOD saves 45.8/31.7 percent of sensors and 33.0/24.0 percent of all nodes compared with CPC/RDS.

From Figs. 8 and 9, we conclude that 1) different object numbers have significant impact on the number of sensors and relay nodes, 2) CPC requires much more nodes compared with other schemes, which shows the necessary of using R&D sensors to reduce the deployment cost, 3) MCD and DOD deploy the minimum number of sensors and relay nodes with different object numbers under the congregating and random distributions of objects, respectively. Below, we compare the performances of RDS, MCD, and DOD since they all allow sensors to rotate.

5.2 Effect of Different Sector Angles θ

We then evaluate the effect of different θ values on the number of sensors deployed by RDS, MCD, and DOD. There are 200 and 400 objects in the sensing field and δ is set to 0.3 and 0.5. Starting from $\theta = 15^\circ$, θ is gradually increased by 15 degrees, until $\theta = 120^\circ$. We measure the number of sensors and their *reduction ratios*, where we take the number of sensors when $\theta = 15^\circ$ as the basis for comparison.

Figs. 10a, 10b, 10c, and 10d show the number of sensors deployed by RDS, MCD, and DOD and their reduction ratios with different θ values under the random distribution of objects. Intuitively, a larger θ value means that a sensor covers a wider range, so more objects can be covered. However, the reduction ratio of RDS is always below 6.5 percent since it arbitrarily selects disks to cover objects. The reduction ratios of MCD and DOD decrease when there are 200 objects. The reason is that when there are fewer objects, they are *sparsely* placed in the sensing field. So, the number of objects covered by each disk may be reduced. Thus, even with a larger θ value, the number of objects covered by each sensor increases slightly. In addition, when $\delta = 0.3$, all schemes have smaller reduction ratios since a smaller δ value helps each sensor cover more sectors in its disk. Therefore, the number of cases to deploy multiple sensors on the same disk may be reduced.

By considering disk overlap, DOD deploys the minimum number of sensors when objects are arbitrarily placed. In addition, it always has the largest reduction ratio since we could easily deploy fewer sensors on the additional disks to cover joint sectors when θ grows. In average, the reduction ratios of RDS, MCD, and DOD are 2.85, 9.15, and 16.75 percent when $\theta = 120^\circ$ (that is, the largest θ value), respectively.

Figs. 10e, 10f, 10g, and 10h show the number of sensors deployed by RDS, MCD, and DOD and their reduction ratios with different θ values under the congregating distribution of objects. We observe that θ has significant impact on the number of sensors. Since objects congregate together, a larger θ value helps a sensor cover more objects. Thus, the number of sensors decreases sharply when θ grows. When there are 200 objects and $\delta = 0.3$, the reduction ratios of all schemes decrease. However, such decreases are not significant as compared with those under the random distribution of objects.

When objects congregate together, MCD deploys the minimum number of sensors. In addition, DOD has the largest reduction ratio since it deploys sensors to cover joint sectors. In average, the reduction ratios of RDS, MCD, and DOD are 33.08, 39.38, and 52.35 percent when $\theta = 120^\circ$, respectively, which are larger than those under the random distribution of objects.

From Fig. 10, we conclude that 1) θ has significant impact on the number of sensors when objects congregate together, 2) DOD always has the largest reduction ratio, and 3) MCD and DOD deploy the minimum number of sensors with different θ values under the congregating and random distributions of objects, respectively.

5.3 Effect of Different δ Values

Finally, we evaluate the effect of different δ values on the number of sensors deployed by RDS, MCD, and DOD.

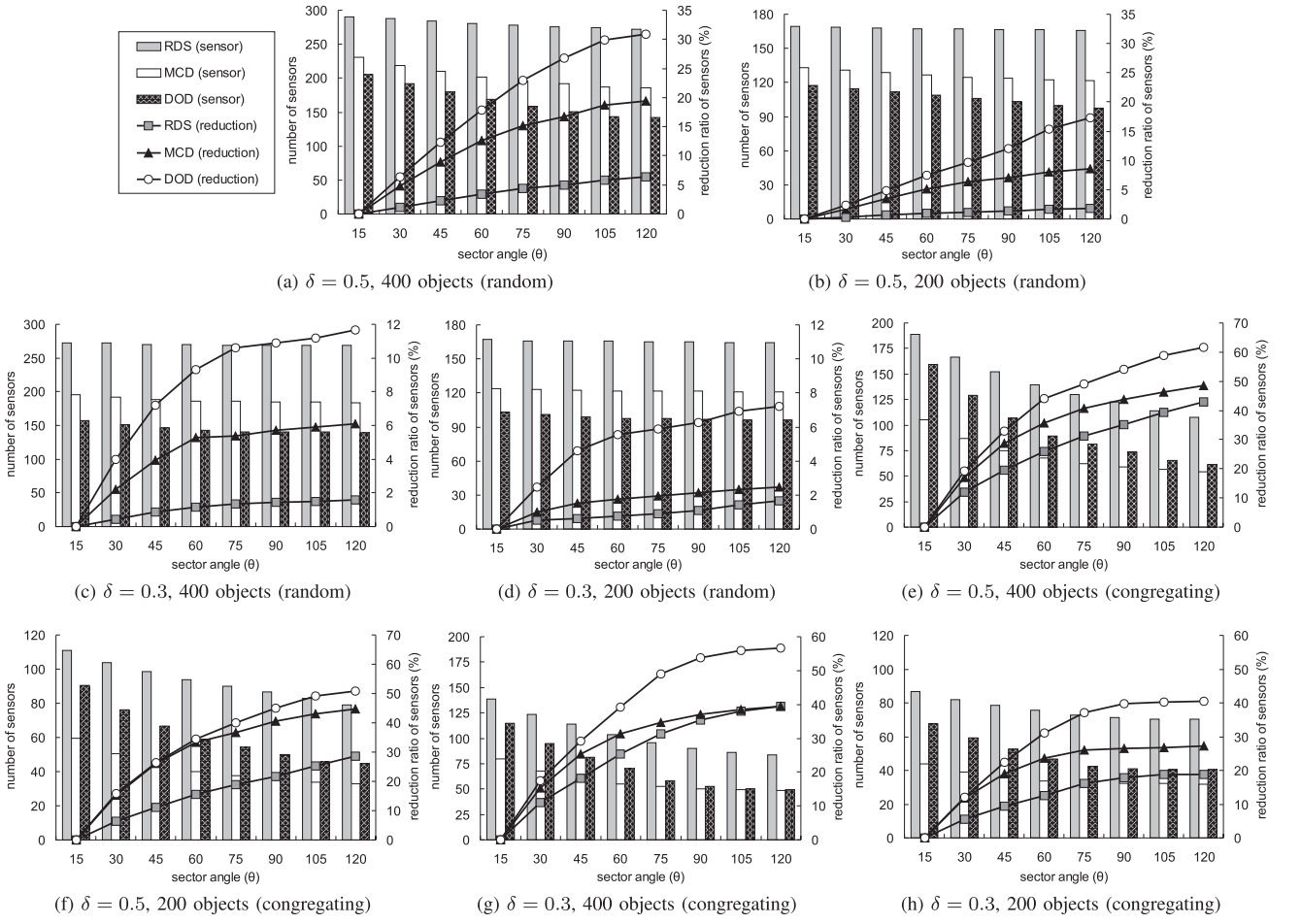


Fig. 10. The number of sensors deployed by RDS, MCD, and DOD with different sector angles θ. (a)-(d) Under the random distribution of objects. (e)-(h) Under the congregating distribution of objects.

There are 200 and 400 objects in the sensing field and we set $\theta = 30^\circ$. The δ value is set to 0.4, 0.3, 0.2, and 0.1, so a sensor can rotate to cover 2, 3, 5, and 10 sectors, respectively. We measure the number of sensors and their reduction ratios, where we take the number of sensors when $\delta = 0.4$ as the basis for comparison.

Figs. 11a and 11b show the number of sensors deployed by RDS, MCD, and DOD and their reduction ratios with different δ values under the random distribution of objects. Intuitively, a smaller δ value means that a sensor covers more sectors, so more sensors are saved. However, the increase of reduction ratio from 0.2 to 0.1 is not significant. The reason is given by considering the following worst case. When δ is 0.4, 0.3, 0.2, and 0.1, each sensor covers totally $2 \times 30^\circ = 60^\circ$, $3 \times 30^\circ = 90^\circ$, $5 \times 30^\circ = 150^\circ$, and $10 \times 30^\circ = 300^\circ$ of angle, so a disk needs at most $\lceil \frac{360^\circ}{60^\circ} \rceil = 6$, $\lceil \frac{360^\circ}{90^\circ} \rceil = 4$, $\lceil \frac{360^\circ}{150^\circ} \rceil = 3$, and $\lceil \frac{360^\circ}{300^\circ} \rceil = 2$ sensors to cover its objects, respectively. Since the difference between the maximum number of sensors used to cover a disk is $3 - 2 = 1$ when δ is 0.2 and 0.1, the increase of reduction ratio is not significant. We observe that DOD always has the largest reduction ratio since it can deploy fewer sensors on the additional disks. In average, the reduction ratios of RDS, MCD, and DOD are 4.7, 11.9, and 21.6 percent when $\delta = 0.1$ (that is, the smallest δ value), respectively.

Figs. 11c and 11d show the number of sensors deployed by RDS, MCD, and DOD and their reduction ratios with different δ values under the congregating distribution of objects. We observe that δ has significant impact on the number of sensors. Since objects congregate together, a smaller δ value allows each sensor to cover more objects. So, the number of sensors decreases sharply when δ reduces. By exploiting disk overlap, DOD always has the largest reduction ratio. In average, the reduction ratios of RDS, MCD, and DOD are 39.1, 40.5, and 54.9 percent when $\delta = 0.1$, respectively, which are larger than those under the random distribution of objects.

From Fig. 11, we conclude that 1) δ has significant impact on the number of sensors, especially when objects congregate together, 2) DOD always has the largest reduction ratio, and 3) MCD and DOD deploy the minimum number of sensors with different δ values under the congregating and random distributions of objects, respectively.

6 Conclusions and Future Work

In this paper, we have defined a temporal coverage model to monitor objects and developed a surveillance system by R&D sensors, which warns users that objects disappear by sending snapshots. We demonstrate our temporal coverage model by the system and point out the NP-hard R&D sensor deployment problem. Two efficient heuristics are

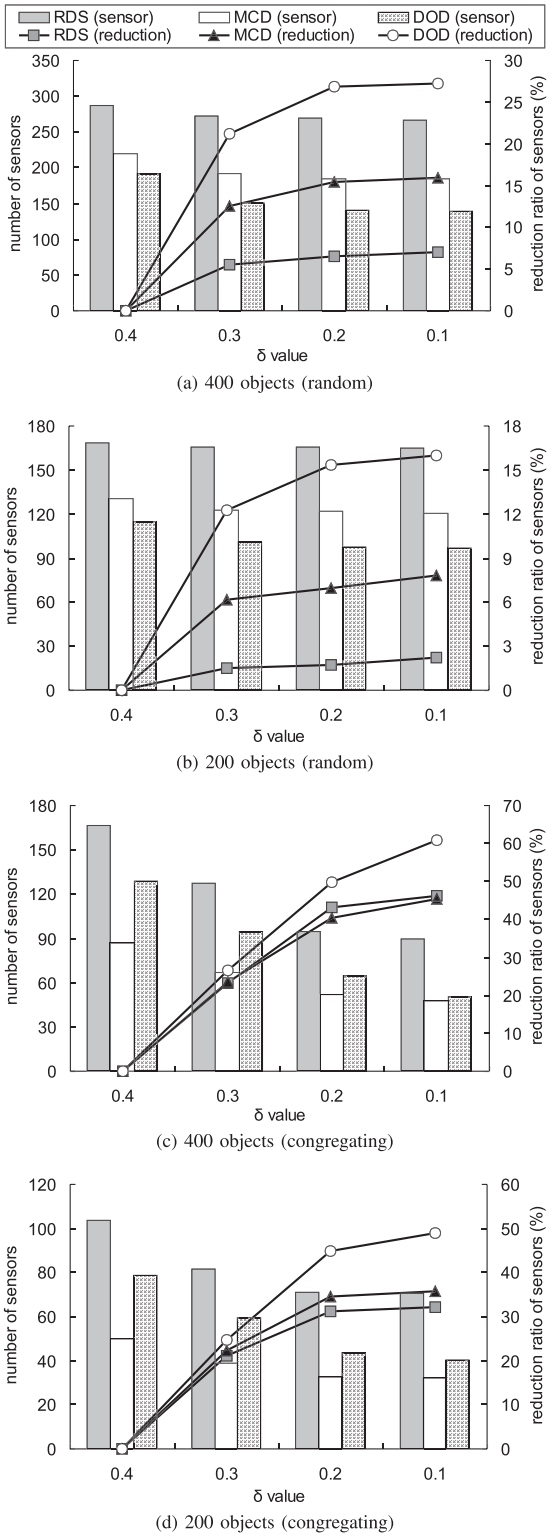


Fig. 11. The number of sensors deployed by RDS, MCD, and DOD with different δ values. (a) and (b) Under the random distribution of objects. (c) and (d) Under the congregating distribution of objects.

proposed, where MCD deploys sensors to cover the disks with more objects, while DOD deploys sensors to cover joint sectors to exploit disk overlap. Simulation results show that MCD performs well when objects congregate together, while DOD works better when objects are arbitrarily placed in the sensing field.

Below, we give future research topics. First, since the R&D sensor deployment problem is NP-hard, we will design more polynomial-time approximate algorithms with lower approximation ratios based on the current result. Second, in this paper, we consider that a sensor takes equal time $\frac{T}{\alpha_i}$ to stay at each sector to detect objects. For some types of sensors, the detection time may depend on the object number, so sensors should stay longer time at the sectors covering more objects. Third, giving the locations of objects and sensors, some studies [42], [51] calculate the directions of sensors to cover objects. These results could help determine the rotation sequences of stepper motors to save their energy and thus extend our deployment heuristics. Finally, using disk overlap and sensor cooperation for fault tolerance deserves further investigation.

REFERENCES

[1] T. Arampatzis, J. Lygeros, and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," *Proc. IEEE Int'l Symp. Mediterranean Conf. Control and Automation Intelligent Control*, pp. 719-724, 2005.

[2] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 6, no. 4, pp. 621-655, 2008.

[3] S. Aoki, J. Nakazawa, and H. Tokuda, "Spinning Sensors: A Middleware for Robotic Sensor Nodes with Spatiotemporal Models," *Proc. IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications*, pp. 89-98, 2008.

[4] R. Horaud, D. Knossow, and M. Michaelis, "Camera Cooperation for Achieving Visual Attention," *Machine Vision and Applications*, vol. 16, no. 6, pp. 1-12, 2006.

[5] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "SensEye: A Multi-Tier Camera Sensor Network," *Proc. 13th Ann. ACM Int'l Conf. Multimedia*, pp. 229-238, 2005.

[6] J.Y. Lee, "Exploiting Constrained Rotation for Localization of Directional Sensor Networks," *Proc. IEEE Int'l Wireless Comm. and Mobile Computing Conf.*, pp. 767-772, 2008.

[7] F. Nussbaum, G.T. Stevens, and J.G. Kelly, "Sensors for a Forward-Looking High Resolution AUV Sonar," *Proc. IEEE Symp. Autonomous Underwater Vehicle Technology*, pp. 141-145, 1996.

[8] I.T. Ruiz, S. de Raucourt, Y. Petillot, and D.M. Lane, "Concurrent Mapping and Localization Using Sidescan Sonar," *IEEE J. Oceanic Eng.*, vol. 29, no. 2, pp. 442-456, 2004.

[9] J.F. Michaels, "An Approach to Radiated Testing of Installed Airborne Doppler Radar with Weather/Windshear Detection Capability," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 12, pp. 25-30, Dec. 1995.

[10] M. Daun, W. Koch, and R. Klemm, "Tracking of Ground Targets with Bistatic Airborne Radar," *Proc. IEEE Radar Conf.*, 2008.

[11] G.K. Das, R. Fraser, A. Lopez-Ortiz, and B.G. Nickerson, "On the Discrete Unit Disk Cover Problem," *Proc. Fifth Int'l Conf. WALCOM: Algorithms and Computation*, pp. 146-157, 2011.

[12] C.F. Huang and Y.C. Tseng, "The Coverage Problem in a Wireless Sensor Network," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519-528, 2005.

[13] G. Simon, M. Molnar, L. Gonczy, and B. Cousin, "Robust K-Coverage Algorithms for Sensor Networks," *IEEE Trans. Instrumentation and Measurement*, vol. 57, no. 8, pp. 1741-1748, Aug. 2008.

[14] L. Liu, H. Ma, and X. Zhang, "On Directional K-Coverage Analysis of Randomly Deployed Camera Sensor Networks," *Proc. IEEE Int'l Conf. Comm.*, pp. 2707-2711, 2008.

[15] S. Kumar, T.H. Lai, and A. Arora, "Barrier Coverage with Wireless Sensors," *Wireless Networks*, vol. 13, no. 6, pp. 817-834, 2007.

[16] L. Zhang, J. Tang, and W. Zhang, "Strong Barrier Coverage with Directional Sensors," *Proc. IEEE Global Telecomm. Conf.*, 2009.

[17] 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.

[18] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak, "Minimal and Maximal Exposure Path Algorithms for Wireless Embedded Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems*, pp. 40-50, 2003.

[19] L. Liu, X. Zhang, and H. Ma, "Minimal Exposure Path Algorithms for Directional Sensor Networks," *Proc. IEEE Global Telecomm. Conf.*, 2009.

[20] C. Gui and P. Mohapatra, "Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks," *Proc. ACM Int'l Conf. Mobile Computing and Networking*, pp. 129-143, 2004.

[21] L. Liu, X. Zhang, and H. Ma, "Exposure-Path Prevention in Directional Sensor Networks Using Sector Model Based Percolation," *Proc. IEEE Int'l Conf. Comm.*, 2009.

[22] C. Gui and P. Mohapatra, "Virtual Patrol: A New Power Conservation Design for Surveillance Using Sensor Networks," *Proc. IEEE Int'l Symp. Information Processing in Sensor Networks*, pp. 246-253, 2005.

[23] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility Improves Coverage of Sensor Networks," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 300-308, 2005.

[24] C.Y. Chang, H.R. Chang, H.J. Liu, and S.W. Chang, "On Providing Temporal Full-Coverage by Applying Energy-Efficient Hole-Movement Strategies for Mobile WSNs," *Proc. IEEE Wireless Comm. and Networking Conf.*, pp. 2778-2783, 2007.

[25] C. Liu and G. Cao, "Spatial-Temporal Coverage Optimization in Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 4, pp. 465-478, Apr. 2011.

[26] K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks," *IEEE Trans. Computers*, vol. 51, no. 12, pp. 1448-1453, Dec. 2002.

[27] S.S. Dhillon and K. Chakrabarty, "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf.*, pp. 1609-1614, 2003.

[28] F.Y.S. Lin and P.L. Chiu, "A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks," *IEEE Comm. Letters*, vol. 9, no. 1, pp. 43-45, Jan. 2005.

[29] 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.*, pp. 114-121, 2005.

[30] X. Bai, S. Kumar, D. Xuan, Z. Yun, and T.H. Lai, "Deploying Wireless Sensors to Achieve Both Coverage and Connectivity," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 131-142, 2006.

[31] T. Sun, L.J. Chen, C.C. Han, and M. Gerla, "Reliable Sensor Networks for Planet Exploration," *Proc. IEEE Int'l Conf. Networking, Sensing and Control*, pp. 816-821, 2005.

[32] Z. Cheng, M. Perillo, and W.B. Heinzelman, "General Network Lifetime and Cost Models for Evaluating Sensor Network Deployment Strategies," *IEEE Trans. Mobile Computing*, vol. 7, no. 4, pp. 484-497, Apr. 2008.

[33] Z. Butler and D. Rus, "Event-Based Motion Control for Mobile-Sensor Networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34-42, Oct.-Dec. 2003.

[34] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces," *Proc. IEEE INFOCOM*, pp. 1293-1303, 2003.

[35] G. Tan, S.A. Jarvis, and A.M. Kermarrec, "Connectivity-Guaranteed and Obstacle-Adaptive Deployment Schemes for Mobile Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 8, no. 6, pp. 836-848, June 2009.

[36] X. Wang and S. Wang, "Hierarchical Deployment Optimization for Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 10, on. 7, pp. 1028-1041, July 2011.

[37] G. Wang, G. Cao, T.L. Porta, and W. Zhang, "Sensor Relocation in Mobile Sensor Networks," *Proc. IEEE INFOCOM*, pp. 2302-2312, 2005.

[38] G. Wang, G. Cao, and T.L. Porta, "Movement-Assisted Sensor Deployment," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 640-652, June 2006.

[39] 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, Feb. 2008.

[40] 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, Sept. 2008.

[41] J. Ai and A.A. Abouzeid, "Coverage by Directional Sensors in Randomly Deployed Wireless Sensor Networks," *J. Combinatorial Optimization*, vol. 11, no. 1, pp. 21-41, 2006.

[42] Y. Cai, W. Lou, M. Li, and X.Y. Li, "Energy Efficient Target-Oriented Scheduling in Directional Sensor Networks," *IEEE Trans. Computers*, vol. 58, no. 9, pp. 1259-1274, Sept. 2009.

[43] H. Ma and Y. Liu, "Some Problems of Directional Sensor Networks," *Int'l J. Sensor Networks*, vol. 2, nos. 1/2, pp. 44-52, 2007.

[44] X. Han, X. Cao, E.L. Lloyd, and C.C. Shen, "Deploying Directional Sensor Networks with Guaranteed Connectivity and Coverage," *Proc. IEEE Conf. Sensor, Mesh and Ad Hoc Comm. and Networks*, pp. 153-160, 2008.

[45] Y.E. Osais, M. St-Hilaire, and F.R. Yu, "Directional Sensor Placement with Optimal Sensing Range, Field of View and Orientation," *Mobile Networks and Applications*, vol. 15, pp. 216-225, 2010.

[46] J. Djugash, S. Singh, G. Kantor, and W. Zhang, "Range-Only SLAM for Robots Operating Cooperatively with Sensor Networks," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 2078-2084, 2006.

[47] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, "iMouse: An Integrated Mobile Surveillance and Wireless Sensor System," *Computer*, vol. 40, no. 6, pp. 60-66, 2007.

[48] Jennic Microcontroller, http://www.jennic.com, 2012.

[49] Zigbee Alliance, "Zigbee Specification: Zigbee Document," 2006.

[50] 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*, pp. 541-546, 2003.

[51] X. Cao, X. Jia, and G. Chen, "Maximizing Lifetime of Sensor Surveillance Systems with Directional Sensors," *Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Networks*, pp. 110-115, 2010.

You-Chiun Wang received the PhD degree in computer science from the National Chiao-Tung University, Taiwan, in 2006. Currently, he is working as an assistant professor in the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. His research interests include wireless communication, mobile computing, and wireless sensor networks. He is a member of the IEEE.

Yung-Fu Chen received the MS degree in computer science from the National Chiao-Tung University, Taiwan, in 2008. His research interest focuses on wireless sensor networks.

Yu-Chee Tseng received the PhD degree in computer and information science from the Ohio State University in 1994. He is a professor (2000-present), chairman (2005-2009), and associate dean (2007-present) in the Department of Computer Science, National Chiao-Tung University, Taiwan. He serves on the editorial boards of *Telecommunication Systems* (2005-present), *IEEE Transactions on Vehicular Technology* (2005-2009), *IEEE Transactions on Mobile Computing* (2006-present), and *IEEE Transactions on Parallel and Distributed Systems* (2008-present). He is a senior member of the IEEE.