Deploying R&D Sensors to Monitor Heterogeneous Objects and Accomplish Temporal Coverage

You-Chiun Wang and Shin-En Hsu

Abstract—A rotatable and directional (R&D) sensor is a wireless device that has sector-shaped sensing coverage and rotatable ability. Such sensors can periodically rotate to monitor nearby objects in order to achieve *temporal coverage*. Specifically, an object is said to be δ_i*-time covered* if it can be monitored by R&D sensors for at least δ_i portion in each period, where $0 < \delta_i \leq 1$. Given a set of heterogeneous objects with different δ_i values, the paper formulates a *generalized R&D sensor deployment (GRSD) problem*. It determines how to use the minimum number of R&D sensors to accomplish temporal coverage by letting each object be δ_i-time covered. The GRSD problem is proven to be NP-hard, and an efficient heuristic is developed based on the locations and coverage demands of objects. Experimental results demonstrate that the proposed GRSD heuristic significantly reduces the number of deployed sensors compared with existing methods, under various simulation scenarios.

Index Terms—directional sensor, node deployment, object monitoring, temporal coverage, wireless sensor network.

1 Introduction

A WIRELESS sensor network consists of many self-configurable, small devices (called sensors) that form an ad hoc network to cooperatively monitor the region of interest. Sensors are usually considered as *omnidirectional*, in the sense that they have disk-shaped sensing coverage used to collect environmental data. They can be also equipped on mobile platforms to tactically move to certain locations to conduct different missions [1]. Recently, wireless sensor networks have been massively applied to various scenarios, such as animal tracking, power management, oceanic exploration, pollution detection, and vehicular safety [2].

Due to their hardware characteristics, some kinds of sensors can monitor data from just one direction. They are generally called *directional sensors*, and practical examples include camera, infrared, light detection and ranging (lidar), sonar, and ultrasonic sensors. Directional sensors have different coverage style with omnidirectional sensors in essence [3], because they can detect only the objects or events located in their *sector-shaped* sensing range. However, in some applications such as searching objects [4], [5], we may ask directional sensors to collect information from multiple or even all directions. To achieve this objective, one possible solution is to install a number of directional sensors on one node, where each sensor faces to a different direction to collect data [6]. However, this solution is not cost-efficient, because we have to use a lot of directional sensors.

An alternative solution is to use some robotic actuators like stepper motors to let directional sensors 'rotate' to detect their surrounding objects or events [7]. This solution not only reduces the number of directional sensors, but also provides spatiotemporal coverage of the environment. In fact, a number of research efforts exploit such *rotatable and directional (R&D) sensors* to develop various applications. For example, the work of [8] employs rotatable airborne radars to estimate the wind velocity. In [5], each robot is equipped with infrared sensors, and it can identify nearby objects (e.g., another robot) by rotating the sensors. Through a rotating array of ultrasonic sensors, the study in [9] proposes a strategy for spatial reconstruction of orthogonal planes. Wang et al. [10] use infrared and camera R&D sensors to develop an object surveillance application.

Motivated by the aforementioned applications, this paper aims at investigating how to accomplish *temporal coverage* by adopting R&D sensors. In particular, we consider that the time axis is divided into repetitive periods. During each period, an R&D sensor can rotate to monitor objects or target locations around it. Notice that the traditional spatial coverage model [11] usually requests sensors to 'always' monitor all objects or target locations. On the contrary, this temporal coverage model allows sensors to monitor different objects or locations at different times, which supports more flexibility.

In this paper, we formulate a *generalized R&D sensor deployment (GRSD) problem* to formally define the temporal coverage model in R&D sensor networks. Suppose that each R&D sensor can rotate 360 degrees, and it spends total (constant) time T to monitor objects in each period. An object is said to be δ_i*-time covered*, $0 < \delta_i \leq 1$, if it is monitored by R&D sensors for at least $\delta_i T$ time in every period. Objects are *heterogeneous*, in the sense that they can have different δ_i values (depending on their importance or the application requirement). Given a set of such objects to be monitored, the GRSD problem determines how to use the minimum number of R&D sensors to cover the objects, so as to satisfy their δ_i-time covered demands.

We use Fig. 1 as an example to illustrate the GRSD problem. Suppose that there are two types of objects needed to be $1/2$-time and $1/4$-time covered, which are denoted by X and Y objects, respectively. Both sectors A and B contain only X objects, so sensor s_i can rotate to cover them, and stop in each sector for $T/2$ time in order to satisfy the coverage requirement of each object. On the other hand, because sector C contains one X object while sectors D and E contain only Y objects,

The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan.
E-mail: ycwang@cse.nsysu.edu.tw; m013040055@student.nsysu.edu.tw

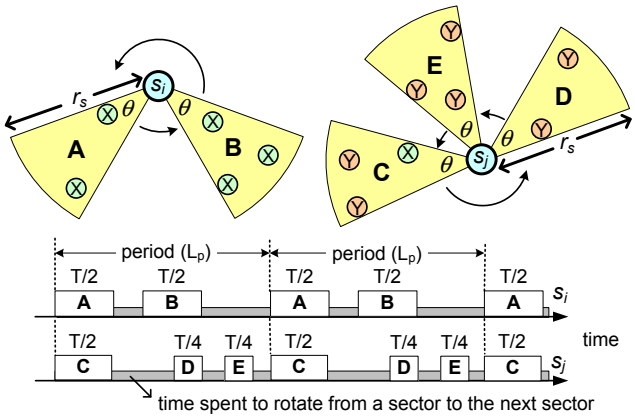


Fig. 1: Achieve temporal coverage by R&D sensors, where X and Y objects need to be $1/2$-time and $1/4$-time covered, respectively.

sensor s_j should spend $T/2$, $T/4$, $T/4$ time to stay in sectors C, D, and E to monitor their objects, respectively. In this case, the coverage requirement of Y objects in sector C can be also met.

In this paper, we prove that the GRSD problem is NP-hard, and develop an efficient heuristic to solve it. The idea is to first compute a set of disks to cover all objects. Then, our heuristic iteratively selects a disk based on the locations and δ_i values of its objects, and places R&D sensors to monitor the objects in that disk. Finally, the deployment result is further improved by exploiting the *residual monitoring time* of R&D sensors and allowing them to 'cooperatively' cover some sectors. Therefore, we can find out potential redundant sensors and remove them accordingly. Extensive simulation results verify that our GRSD heuristic can use a smaller number of R&D sensors to provide temporal coverage of heterogeneous objects compared with other deployment methods. This paper contributes in 1) defining a temporal coverage model by R&D sensors, 2) formulating an R&D sensor deployment problem that considers heterogeneous objects, 3) verifying the NP-hard property of the deployment problem, and 4) developing an efficient heuristic to reduce the network deployment cost.

The rest of this paper is organized as follows: The next section gives related work. Section 3 formulates the GRSD problem, and our heuristic to the problem is presented in Section 4. Section 5 evaluates the performance of different deployment methods by simulations. Finally, the conclusion is drawn in Section 6.

2 Related Work

Traditional sensor coverage problems usually aim at providing *spatial coverage* for a sensing field or a long-thin barrier [11]. Therefore, in this section we focus our discussion on *temporal coverage* by omnidirectional sensors. Then, we survey the coverage and deployment schemes in directional sensor networks.

2.1 Temporal Coverage by Omnidirectional Sensors

Several studies consider how to achieve temporal coverage by static, omnidirectional sensors. For example, the work of [12] activates only a small number of sensors at any time to monitor the sensing field. These active sensors can form an *active zone* to monitor objects. As time goes by, the active zone will move along a certain trajectory to conduct the monitoring job. Liu and Cao [13] partition the sensing field into multiple subareas. They define a *spatial-temporal coverage metric* for each subarea, which is the product of the subarea's size and the period that the subarea is covered. Then, the objective is to turn on a subset of sensors at different times, such that the metric sum of all subareas can be maximized.

Mobile, omnidirectional sensors are also widely used to support temporal coverage. A lot of research addresses how to dynamically move sensors to cover different sensing fields. Both [14] and [15] imagine that sensors can exert virtual forces on each other to move. Thus, they can be uniformly distributed over the sensing field. Given the initial (random) deployment of sensors, the studies of [16], [17] use Voronoi diagrams to find out coverage holes, and then move sensors to eliminate these holes. The work of [18] proposes a two-phase strategy to maneuver mobile sensors to cover a sensing field. The first phase computes where sensors should be placed in the sensing field. Then, the second phase dispatches sensors to the above locations, such that they can spend less energy on movement. In [19], Wang and Tseng develop a distributed solution to move sensors, so as to provide k-coverage of a sensing field.

Some research efforts tactically dispatch mobile sensors to visit event locations. The work of [20] assumes that events only appear in certain positions and their arrival/departure time distribution is known in advance. Then, the work deals with the problem of using the minimum number of sensors (and calculating their moving trajectories) to reduce the event loss probability. On the other hand, Wang et al. [21] consider that events could arbitrarily appear in the sensing field. Given the locations of event occurrence, they investigate how to dispatch mobile sensors to visit these locations, such that the lifetime of mobile sensors can be maximized. The work of [22] assumes that mobile sensors have multiple capabilities, so they can analyze different types of events. Then, the work develops an energy-efficient algorithm that dispatches mobile sensors with different capabilities to visit the locations of heterogeneous events.

Chang et al. [23] consider a mobile sensor network with a large coverage hole, but the number of sensors is not sufficient to fill that hole. Then, they move a subset of sensors to make the hole 'migrate' in the sensing field. Therefore, every location can be eventually covered by sensors for a threshold time. Given a set of points, the work of [24] periodically moves sensors to cover them. A point is called t_i-*sweep covered* if it is covered by a sensor in each sweep period t_i. Then, the work solves the *min-sensor sweep coverage problem*, which asks how to use the minimum number of sensors to make every point be t_i-sweep covered.

We can observe that a variety of temporal coverage issues have been addressed in omnidirectional sensor networks. However, they have not been well investigated in directional sensor networks. Therefore, it motivates us to study temporal coverage by R&D sensors in this paper.

2.2 Coverage and Deployment Schemes in Directional Sensor Networks

A number of research efforts address the coverage issue by using directional sensors. Given a set of objects, Ai and Abouzeid [25] formulate an integer linear programming that uses the minimum number of directional sensors to cover the maximum number of objects. Then, a greedy-based solution is developed to schedule the wake-up period of sensors. Cai et al. [26] organize directional sensors into multiple *cover sets*

according to their facing directions, where the sensors in a cover set can monitor all objects in the sensing field. Then, they investigate how to activate only one cover set of sensors in different periods to extend the system lifetime. The study of [27] considers that objects have different requirements of coverage quality. It then organizes the directions of sensors into a group of non-disjoint sets, each being able to meet the coverage quality of all objects. Obviously, these studies have different objectives with our work.

How to deploy static, directional sensors is also discussed in the literature. Han et al. [28] propose two deployment problems for directional sensors. The *connected region-coverage problem* asks how to use the minimum number of directional sensors to cover an infinite plane. It can be solved by continuously placing sectors in a hexagon-like fashion on the plane. Then, the *connected point-coverage problem* determines how to use the minimum number of directional sensors to cover a given set of point-locations. It can be solved by searching the sectors anchored by one, two, and three point-locations that cover the most point-locations, and then greedily deploying sensors to cover them. On the other hand, the work of [29] models the sensing field by a set of points on which directional sensors can be deployed. Given a set of *critical points*, the work adopts the integer linear programming to compute the minimum number of directional sensors to monitor all critical points. Tao and Wu [30] survey the solutions of different barrier coverage problems by using directional sensors. Nevertheless, the aforementioned studies cannot support temporal coverage, because directional sensors are fixed after deployment.

Some work exploits the rotatable ability of R&D sensors. The study of [31] assumes that R&D sensors can rotate within a limited angle. In addition, the sensing field contains some *anchor sensors* with known positions. Then, the work solves the localization problem by calibrating the positions of non-anchor sensors. Specifically, each of such sensors measures the relative ranges from neighboring anchor sensors. Then, the sensor compensates the confined field of view by rotating. Obviously, this work does not aim at the deployment issue. On the other hand, the study of [10] proposes an *R&D sensor deployment problem* that asks how to use the minimum number of R&D sensors to guarantee that every object is δ-time covered, where $0 < \delta \leq 1$. Two deployment methods are developed to solve the problem. The *maximum covering deployment (MCD) method* iteratively places an R&D sensor to cover more objects, while the *disk-overlapping deployment (DOD) method* places R&D sensors to cover the joint sectors of overlapped disks formed by the rotation of sensors. In fact, this R&D sensor deployment problem is a special case of our GRSD problem, because it assumes that all objects have the same δ value. Compared with [10], our work formulates a more 'generalized' R&D sensor deployment problem by taking object heterogeneity into account. Through simulations, we will also compare our GRSD heuristic with both MCD and DOD methods in Section 5.

3 The Generalized R&D Sensor Deployment (GRSD) Problem

Given a set of fixed objects $\hat{\mathbf{O}} = \{o_1, o_2, \cdots, o_m\}$, our objective is to deploy R&D sensors to support temporal coverage for them. Each sensor has the sensing range modeled by a sector with an opening angle of $\theta \in (0, \pi)$ and a radius of r_s. Besides, its communication range is modeled by a disk with a radius of r_c. Sensors are *homogeneous*, in the sense that they have the same θ, r_s, and r_c values. However, the relationship between r_s and r_c is arbitrary. We adopt the *binary sensing model*, where an object is assumed to be covered by a sensor if it is inside the sensing coverage of that sensor. However, our deployment result can be easily applied to the *probabilistic sensing model* by giving a threshold detection probability p_{th}. Specifically, the probability that an object $o_i \in \hat{\mathbf{O}}$ can be detected by a sensor s_j is formulated by [18]:

$$Pr(o_i, s_j) = \begin{cases} e^{-\xi \cdot l(o_i, s_j)} & \text{if } l(o_i, s_j) \leq r_s \\ 0 & \text{otherwise,} \end{cases}$$

where ξ is a parameter used to represent the physical characteristics of a sensor, and $l(\cdot, \cdot)$ denotes the distance function. To let every object be detected by a sensor with the probability no less than p_{th}, we can 'shrink' the sensing distance r_s by

$$Pr(o_i, s_j) = e^{-\xi r_s} \geq p_{th} \Rightarrow r_s \leq -\frac{\ln p_{th}}{\xi}. \quad (1)$$

Each sensor has 360 degrees of freedom to rotate, and it can stop to monitor objects during rotation. Assume that all sensors keep rotating in the same direction (e.g., counterclockwise), and the rotation speed is a constant, say, v degrees per second. Then, the time axis can be divided into multiple *periods* with length L_p, during which a sensor finishes rotating 360 degrees. In particular, we have

$$L_p = t_j^{monitor} + \frac{360}{v},$$

where $t_j^{monitor}$ is the total time that a sensor s_j stops to monitor the objects in a period. Since every sensor has the same L_p and v values, $t_j^{monitor}$ should be fixed for all sensors. Thus, we replace $t_j^{monitor}$ by a constant T. Then, we define that an object $o_i \in \hat{\mathbf{O}}$ is *δ_i-time covered* if o_i can be monitored by R&D sensors for at least $\delta_i T$ time in every period, where $0 < \delta_i \leq 1$. Fig. 1 gives an example, where the objects in sectors A, B, and C are 1/2-time covered, and the objects in sectors D and E are 1/4-time covered.

Suppose that each object $o_i \in \hat{\mathbf{O}}$ is modeled by a point-location and needs to be δ_i-time covered. The GRSD problem then asks how to deploy the minimum number of R&D sensors and determine their *rotation schedules*, such that the coverage requirements of all objects in $\hat{\mathbf{O}}$ can be satisfied. An example is given in Fig. 1, where sensors s_i and s_j have rotation schedules $\{(A, T/2), (B, T/2)\}$ and $\{(C, T/2), (D, T/4), (E, T/4)\}$, respectively. Theorem 1 proves the NP-hard property of the GRSD problem.

Theorem 1. The GRSD problem is NP-hard.

Proof: To prove that the GRSD problem is NP-hard, we reduce an NP-complete problem, the *geometric disk cover (GDC) problem* [32], to one of its instances. The GDC problem determines how to place the minimum number of disks to cover a set of point-locations. Therefore, we construct a GRSD problem instance as shown in Fig. 2. Specifically, the sensing field is divided into two parts, where they are separated by a distance of $2r_s$. There are two types of objects, say, X and Y objects that respectively require δ_X-time and δ_Y-time covered, where $\delta_X = \theta/2\pi$, $\delta_Y = \theta/\pi$, and π is divisible by θ. The left (respectively, right) part of the sensing field contains only X (respectively, Y) objects. Then, we prove that the GRSD problem has a solution if and only if the GDC problem has a solution.

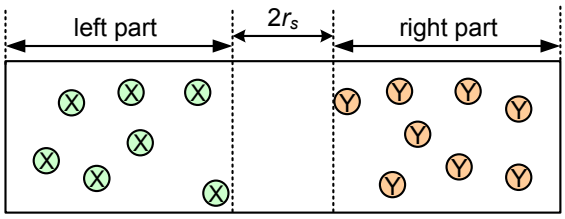


Fig. 2: A GRSD problem instance, where X and Y objects require $\theta/2\pi$-time and θ/π-time covered, respectively.

Suppose that we have a solution to the GDC problem, which is a set of disks $\hat{\mathbf{D}}$. Since the two parts of the sensing field is separated by a distance of $2r_s$, it is impossible that there exists a disk in $\hat{\mathbf{D}}$ that can cover the objects in both parts. Therefore, we can separately discuss the disks in each part. For the left part, since $\delta_X = \theta/2\pi$, each R&D sensor can rotate to cover $2\pi/\theta$ sectors in every period. Because each sector has the opening angle of θ, these sectors must form a complete disk. In other words, an R&D sensor can rotate to cover all objects in its disk per period. Therefore, for each disk of $\hat{\mathbf{D}}$ in the left part, we can deploy an R&D sensor on its center. Similarly, for the right part, since $\delta_Y = \theta/\pi$, each R&D sensor can rotate to cover π/θ sectors in every period. In this case, two R&D sensors (located at the same position) together can rotate to cover a whole disk. Therefore, we can deploy at most two R&D sensors on the center of each disk of $\hat{\mathbf{D}}$ in the right part. These R&D sensors in both parts constitute a solution to the GRSD problem, which proves the *if* statement.

Conversely, suppose that we have a solution to the GRSD problem, which is a set of R&D sensors $\hat{\mathbf{S}}$. Again, we can separately discuss the R&D sensors of $\hat{\mathbf{S}}$ in each part of the sensing field, because it is impossible that there exists a sensor in $\hat{\mathbf{S}}$ that can cover the objects in both parts. For the left part, every R&D sensor can rotate to cover all objects around it, which forms a disk. We thus place a disk whose center is located on each R&D sensor of $\hat{\mathbf{S}}$ in the left part. On the other hand, there are two cases in the right part. The first case is that a single R&D sensor can rotate to cover all objects around it. In this case, we place a disk such that its center is at the sensor's location. The second case is that two R&D sensors are deployed on the same location to cover all objects around them (here, each sensor can rotate to cover at most π/θ sectors, which forms a half disk). Therefore, we place a disk whose center is at the location of both sensors. The above disks in both parts constitute a solution to the GDC problem, thereby proving the *only if* statement. □

4 The Proposed GRSD Heuristic

Given the locations and δ_i values of m objects in $\hat{\mathbf{O}}$, Fig. 3 presents the flowchart of our GRSD heuristic. It consists of the following five steps:

- **[Step 1] Finding disks:** By taking the locations of objects as the input, this step will calculate a set of disks $\hat{\mathbf{D}}$ to cover all objects in $\hat{\mathbf{O}}$, where each disk in $\hat{\mathbf{D}}$ has a radius of r_s. However, if the probabilistic sensing model is considered, we should adjust the value of r_s according to Eq. (1).
- **[Step 2] Computing sectors:** For each disk in $\hat{\mathbf{D}}$, we cut it into *non-overlapping sectors* based on the distribution of objects in that disk. All sectors have an opening angle of θ, and they indicate where R&D sensors should rotate to monitor objects.
- **[Step 3] Placing sensors:** We then iteratively place an R&D sensor to cover a number of sectors in one disk, until all objects in $\hat{\mathbf{O}}$ are covered by sensors. To reduce the number of sensors used in this step, we should take the δ_i values of objects into consideration.
- **[Step 4] Removing redundancy:** When a disk is placed with multiple R&D sensors, we can check whether these sensors have *residual monitoring time*. If so, we can ask two or more sensors to cooperatively cover a sector by combining their residual monitoring time. In this case, some sensors will become 'redundant', if they no longer cover any sector. Therefore, we can remove them to further save the network deployment cost.
- **[Step 5] Connecting nodes:** The aforementioned R&D sensors may not necessarily form a connected network. In this case, we have to add relay nodes to maintain the network connectivity.

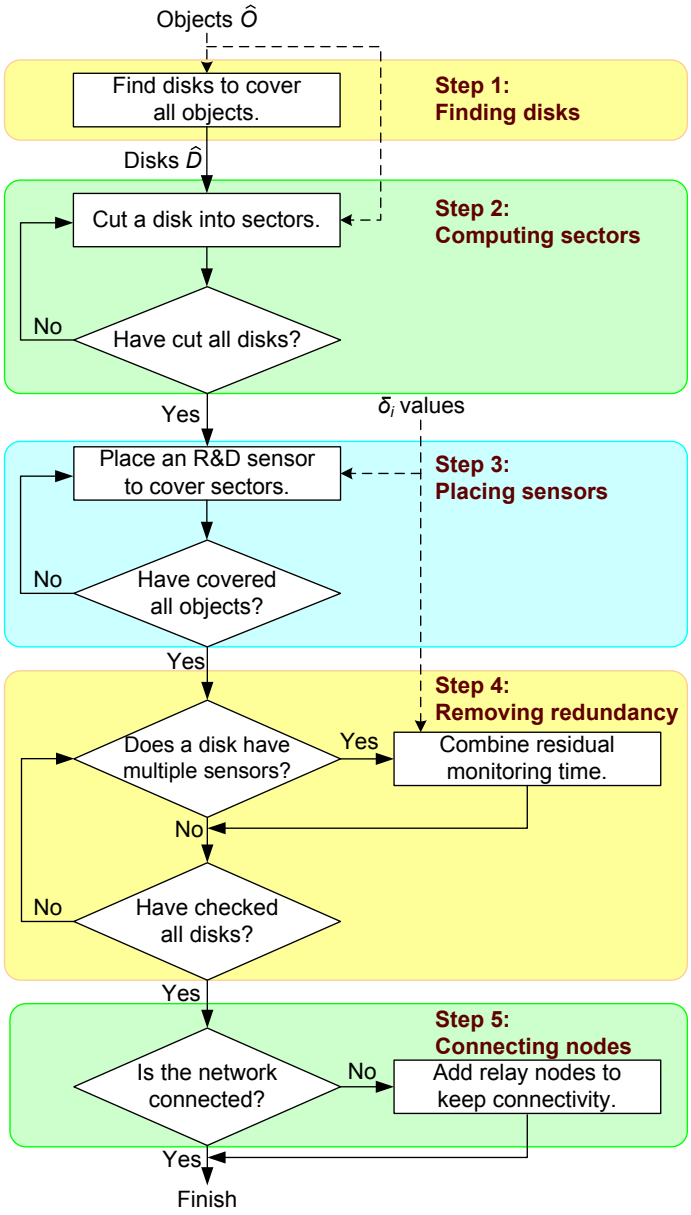


Fig. 3: The flowchart of our proposed GRSD heuristic.

TABLE 1: Summary of notations.

notation	definition
$\hat{\mathbf{O}}$	the set of m objects that need to be monitored, where each object $o_i \in \hat{\mathbf{O}}$ is a point-location
$\hat{\mathbf{D}}$	the set of disks that can cover all objects in $\hat{\mathbf{O}}$, where each disk $d_j \in \hat{\mathbf{D}}$ has a radius of r_s
$\hat{\mathbf{D}}_{\mathbf{L}}$	the subset of disks that have objects with the largest δ_i values
δ_i	the coverage requirement of an object o_i in $\hat{\mathbf{O}}$, where o_i should be covered for $\delta_i T$ time in a period, and $0 < \delta_i \leq 1$
δ_j^{sum}	the sum of δ_i values of all objects in a disk d_j
δ_i^{max}	the largest δ_i value of objects covered by a sensor
δ_i^{sc}	the largest δ_i value of objects in a sector
$T_i^{\mathbf{o}}$	the occupied monitoring time of a sensor s_i
$T_i^{\mathbf{r}}$	the residual monitoring time of a sensor s_i

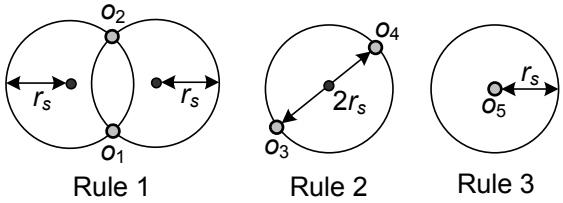


Fig. 4: Three rules to find a set of disks $\hat{\mathbf{D}}$ to cover all objects.

Next, we discuss the detailed design of each step. Table 1 summarizes the common notations used in the GRSD heuristic.

4.1 Step 1: Finding Disks

The objective of this step is to efficiently find a set of disks $\hat{\mathbf{D}}$ to cover the objects in $\hat{\mathbf{O}}$. To do so, we develop three rules to calculate the set $\hat{\mathbf{D}}$ based on the GDC scheme in [32]:

- **Rule 1:** If $l(o_i, o_j) < 2r_s$, where o_i and $o_j \in \hat{\mathbf{O}}$, we place two disks such that their peripheries intersect at both o_i and o_j.
- **Rule 2:** If $l(o_i, o_j) = 2r_s$, we place one disk such that its periphery passes both o_i and o_j.
- **Rule 3:** If $l(o_i, o_k) > 2r_s$ for any $o_k \in \{\hat{\mathbf{O}} - o_i\}$, which means that o_i is *isolated*, we place one disk whose center is at o_i's position.

Fig. 4 gives an example to illustrate the above three rules. Here, we may calculate two disks according to each pair of objects in $\hat{\mathbf{O}}$ (by rule 1), so the maximum number of disks in $\hat{\mathbf{D}}$ will be

$$|\hat{\mathbf{D}}| = 2 \cdot C(m, 2) = \frac{2 \cdot m!}{(m-2)! \cdot 2!} = m(m-1) = O(m^2).$$

Obviously, $|\hat{\mathbf{D}}|$ is much larger than the number of objects in $\hat{\mathbf{O}}$ (i.e., m). It indicates that a lot of disks in $\hat{\mathbf{D}}$ are actually unnecessary. Thus, we only select at most $O(m)$ *candidate disks* in $\hat{\mathbf{D}}$ and remove others. In particular, a candidate disk should have two properties:

1) The disk covers the maximum number of objects.
2) The disk can cover more objects with larger δ_i values.

To do the selection, we set the status of each object in $\hat{\mathbf{O}}$ to *unchecked*. Let us denote by

$$\delta_j^{sum} = \sum\{\delta_i \mid \text{object } o_i \text{ is unchecked and in disk } d_j \in \hat{\mathbf{D}}\}.$$

In other words, δ_j^{sum} is the sum of δ_i values of all unchecked objects in disk d_j. Then, we iteratively select the disk with the maximum δ_j^{sum} value as a candidate disk, and mark all objects in that disk as *checked*. This iteration is repeated until all objects in $\hat{\mathbf{O}}$ become checked. Here, our idea is based on the observation that selecting the disk with the maximum δ_j^{sum} value can have a larger opportunity that the disk covers more objects, and these objects also have larger δ_i values. This method can help reduce the computation cost. Notice that each object is included in at most one candidate disk, so we have $|\hat{\mathbf{D}}| = O(m)$. Lemma 1 gives the time complexity of step 1.

Lemma 1. Finding the set of disks $\hat{\mathbf{D}}$ requires at most $O(m^2)$ time.

Proof: We first apply the three rules to find all disks in $\hat{\mathbf{D}}$. Obviously, these three rules require to check every *possible* pair of objects in $\hat{\mathbf{O}}$. Because we have $|\hat{\mathbf{O}}| = m$, it thus takes $O(C(m, 2))$ time to do the check. Then, we iteratively select the disk that has the largest δ_j^{sum} value to be a candidate disk, which can be implemented by using a maximum binary heap. Because the set $\hat{\mathbf{D}}$ can contain at most $O(2C(m, 2))$ disks, constructing the maximum binary heap (to maintain all disks in $\hat{\mathbf{D}}$) will take time of $O(2C(m, 2))$. In addition, it spends time of $O(\lg 2C(m, 2))$ to extract the maximum value from the heap. Since there are m objects in $\hat{\mathbf{O}}$, it is impossible to do more than $O(m)$ extracting operations from the heap. Thus, finding all candidate disks requires time of $O(m) \cdot O(\lg 2C(m, 2))$. By taking the sum of the above calculation, we can derive the overall computation time by

$$O(C(m, 2)) + O(2C(m, 2)) + O(m) \cdot O(\lg 2C(m, 2)) = O(m^2).$$

□

4.2 Step 2: Computing Sectors

After calculating $\hat{\mathbf{D}}$, we then cut every disk in $\hat{\mathbf{D}}$ into sectors such that 1) these sectors are not overlapped with each other, 2) they contain all objects in the disk, and 3) the number of sectors is minimized. Here, each sector actually points out where an R&D sensor has to rotate to cover the objects in the corresponding disk. Thus, the above three objectives guarantee that we can deploy the minimum number of R&D sensors in each disk of $\hat{\mathbf{D}}$.

The work of [10] proposes a *sector cutting (SC) operation* to compute sectors. It starts by arbitrarily indexing an object as o_1, and adds o_1 to a cluster 1. Then, the SC operation scans other non-indexed objects from o_1 in a counterclockwise direction. Once finding a non-indexed object, it is indexed by o_2. Then, the SC operation decides what cluster o_2 should belong to. Specifically, if $\angle o_1 c_j o_2 \leq \theta$, o_2 belongs to cluster 1; otherwise, o_2 is added to a new cluster 2, where c_j is the disk's center. Similarly, suppose that an object is indexed by o_i, and it belongs to cluster k. Then, the next non-indexed object is indexed by o_{i+1}. If $\angle o_i c_j o_{i+1} \leq \theta$, o_{i+1} belongs to cluster k; otherwise, it is added to a new cluster $k+1$. For example, three clusters are found in Fig. 5(a). Since the included angle between cluster 1 and cluster 3 (i.e., the last

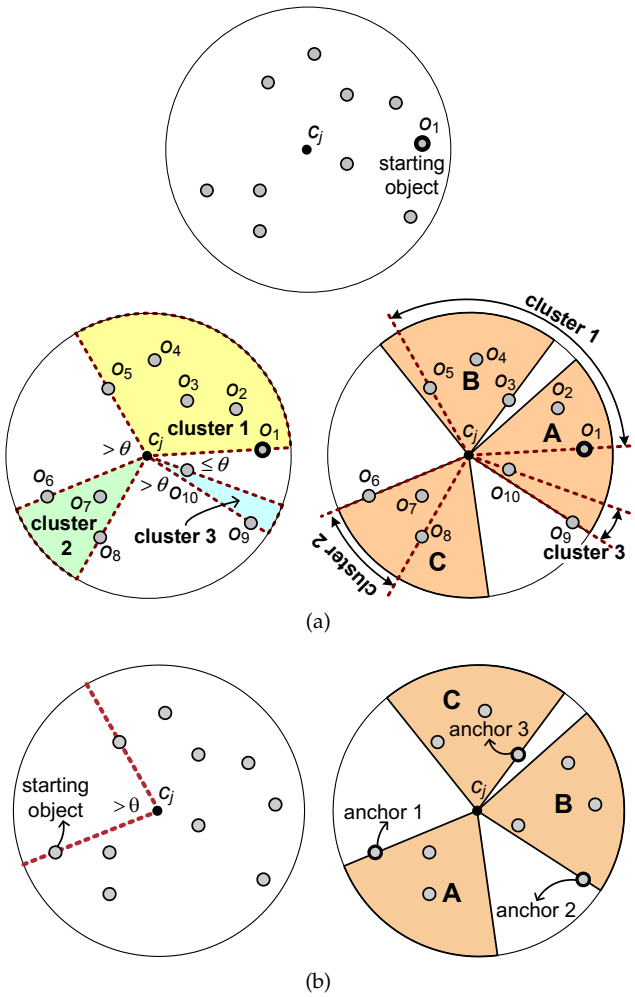


Fig. 5: Two schemes to find sectors in each disk: (a) the SC operation in [10] and (b) our ASF operation.

cluster) is smaller than θ, the SC operation then starts placing sectors from cluster 3 in a counterclockwise direction. For each cluster, the SC operation iteratively places a sector whose right edge passes the *uncovered* object with the smallest index, until all objects in the cluster have been covered. Fig. 5(a) gives an example, where three sectors A, B, and C are calculated by the SC operation.

However, the SC operation requires two 'complete' rounds to scan all objects in a disk (one is to cluster objects, and the other is to place sectors), which is complicated. Therefore, we propose an *anchor-based sector finding (ASF) operation* to reduce the computation cost. Specifically, we first find two adjacent objects, say, o_i and o_{i+1}, such that $\angle o_i c_j o_{i+1} > \theta$. In this case, o_{i+1} will be an *anchor*. In case that we cannot find such two objects, we randomly pick one object as the anchor. Starting from this anchor, we place a sector to let its right edge pass the anchor. Then, the first uncovered object (in the counterclockwise direction) will be the new anchor, and we can place a sector accordingly. We repeat this iteration until all objects in the disk become covered. Fig. 5(b) shows an example, where three anchors are found to help place sectors. Comparing with the SC operation, our ASF operation does not require to index objects. Thus, the ASF operation is simpler than the SC operation. Theorem 2 proves that the ASF operation is an optimal solution. In addition, Lemma 2 analyzes the computation cost of step 2.

Theorem 2. The ASF operation can always find the minimum number of sectors to cover all objects in each disk.

Proof: The SC operation is shown to be optimal in [10]. Thus, we prove that our ASF operation can find the equal number of sectors with the SC operation. There are two cases to be discussed:

Case 1: The included angle between any two adjacent objects is no larger than θ. Thus, there must be only one cluster in the disk. In this case, because both the SC and ASF operations start placing sectors from a random object in a counterclockwise direction, they must calculate the equal number of sectors for the disk.

Case 2: Some adjacent objects have included angles larger than θ. Thus, there could be multiple clusters in the disk. If the ASF operation can always select the first object of a cluster (i.e., the object with the smallest index in the SC operation) as an anchor to place sectors, then it means that the ASF operation must find the same set of clusters with the SC operation. Therefore, we show that the first object of every cluster in the SC operation must be also an anchor in the ASF operation. Without the loss of generality, we assume that the included angle between the first and last clusters is larger than θ.[1] We prove this argument by contradiction. In particular, suppose that o_i and o_{i+1} are the last and first objects in two clusters k and $k+1$ in the SC operation, respectively, but o_{i+1} is not an anchor in the ASF operation. It means that the ASF operation has found a sector that covers both objects o_i and o_{i+1}. However, since o_i and o_{i+1} belong to different clusters, $\angle o_i c_j o_{i+1}$ must be larger than θ, which obviously results in a contradiction. So, the first object indexed by the SC operation in every cluster must be also an anchor in the ASF operation. Therefore, the ASF operation has found the same clusters with the SC operation. In addition, because the SC and ASF operations place the sectors in each cluster by the same manner, they must calculate the equal number of sectors in a disk. Therefore, the ASF operation is an optimal solution. □

Lemma 2. It takes no more than $O(2m)$ time to compute the sectors of all disks in $\hat{\mathbf{D}}$ by the ASF operation.

Proof: The worst case of the ASF operation occurs when every two adjacent objects in each disk have an included angle no larger than θ. In this case, the ASF operation will scan all objects in each disk once, and then randomly pick one object to be the first anchor. Then, it places sectors in the disk by scanning all objects in the counterclockwise direction. Thus, the ASF operation has to scan all objects in each disk twice in the worst case. Because each object in $\hat{\mathbf{O}}$ is included in at most one disk in $\hat{\mathbf{D}}$ (by step 1), it means that the total number of objects scanned by the ASF operation (in one time) will be m. Therefore, it takes $O(m+m) = O(2m)$ time for the ASF operation to find all sectors in the worst case. □

4.3 Step 3: Placing Sensors

In this step, we iteratively place an R&D sensor to cover some sectors of a disk. Here, the maximum number of sectors that can be covered by one R&D sensor depends on the *largest* δ_i value of objects in these sectors. Specifically, the R&D sensor can rotate to cover at most $\lfloor 1/\delta_i^{max} \rfloor$ sectors, where δ_i^{max} is the largest δ_i value of objects covered by the sensor. (How to relax this assumption will be discussed later.) Therefore, we

1. In fact, that is why the SC operation starts placing sectors from the last cluster. If the first and last clusters have an included angle no larger than θ, they are treated as the same cluster in the SC operation.

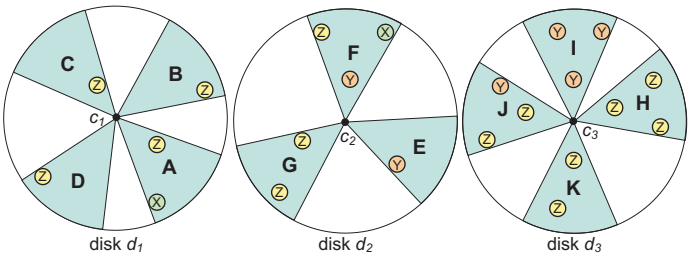


Fig. 6: An example of step 3 in the GRSD heuristic, where X, Y, and Z objects require 0.7-time, 0.5-time, and 0.3-time covered, respectively.

should first deal with those disks containing the objects with the largest δ_i value, because they are critical in terms of the number of sectors that can be covered by each R&D sensor. To do so, we set the status of each object in $\hat{\mathbf{O}}$ to *unchecked* again. Then, we select the subset of disks $\hat{\mathbf{D}}_\mathbf{L} \subseteq \hat{\mathbf{D}}$ that have unchecked objects with the largest δ_i value. For each disk d_j in $\hat{\mathbf{D}}_\mathbf{L}$, we calculate the maximum number of unchecked objects that can be covered by one R&D sensor, which is denoted by u_j. In particular, if d_j has more than $\lfloor 1/\delta_i \rfloor$ sectors, we sort all sectors of d_j by the number of unchecked objects and their δ_i values in a decreasing order. Thus, u_j will be the number of unchecked objects in the first $\lfloor 1/\delta_i \rfloor$ sectors. Otherwise, u_j is the total number of unchecked objects in d_j. We then select the disk d_j in $\hat{\mathbf{D}}_\mathbf{L}$ that has the largest u_j value, and deploy an R&D sensor at its center to cover the sectors accordingly. When there is a tie, we select the disk that contains the maximum number of uncovered objects with the largest δ_i value. Suppose that the R&D sensor covers k sectors in the disk, where $k \leq \lfloor 1/\delta_i \rfloor$. Then, the sensor has to monitor each sector for T/k time in every period. Also, we set these covered objects to *checked*. The above iteration is repeated until all objects in $\hat{\mathbf{O}}$ become checked.

We use the example in Fig. 6 to illustrate the operation in step 3. Suppose that $\hat{\mathbf{D}} = \{d_1, d_2, d_3\}$ and the sensing field contains X, Y, and Z objects, where $\delta_X = 0.7$, $\delta_Y = 0.5$, and $\delta_Z = 0.3$, respectively. By using the ASF operation, there are 11 sectors (denoted by A to K) found in the disks. Then, we place R&D sensors according to the following iterations:

1) We first check those disks containing X objects, so $\hat{\mathbf{D}}_\mathbf{L} = \{d_1, d_2\}$. In this case, only sectors A and F have X objects, and a sensor can cover $\lfloor 1/\delta_X \rfloor = \lfloor 1/0.7 \rfloor = 1$ sector. Since sector F has three objects, but sector A has two objects, we thus place a sensor s_1 on c_2 to cover sector F.
2) Because only d_1 remains in $\hat{\mathbf{D}}_\mathbf{L}$ (for X objects), we thus place a sensor s_2 on c_1 to cover sector A.
3) We then check the disks containing Y objects, so $\hat{\mathbf{D}}_\mathbf{L} = \{d_2, d_3\}$. In this case, a sensor can rotate to cover $\lfloor 1/\delta_Y \rfloor = \lfloor 1/0.5 \rfloor = 2$ sectors. For d_2, a sensor can cover sectors E and G, which totally have three objects (i.e., $u_2 = 3$). For d_3, there are three cases that a sensor can cover the maximum number of objects (where $u_3 = 6$):
   - Case 1: sectors H and I, with three Y objects.
   - Case 2: sectors H and J, with one Y object.
   - Case 3: sectors I and J, with four Y objects.

   Because $u_3 > u_2$, we only consider the three cases in d_3. Here, we choose case 3, because the sensor can cover the maximum number of Y objects. Therefore, we place a sensor s_3 on c_3 to cover both sectors I and J.
4) Since $\hat{\mathbf{D}}_\mathbf{L} = \{d_2\}$, we thus place a sensor s_4 on c_2 to cover both sectors E and G.
5) We then check the sectors containing Z objects, so $\hat{\mathbf{D}}_\mathbf{L} = \{d_1, d_3\}$. In this case, a sensor can rotate to cover $\lfloor 1/\delta_Z \rfloor = \lfloor 1/0.3 \rfloor = 3$ sectors. For d_1, a sensor can rotate to cover sectors B, C, and D, which have totally three objects (i.e., $u_1 = 3$). On the other hand, although only two sectors H and K remain in d_3, they contain totally five objects (i.e., $u_3 = 5$). Thus, we place a sensor s_5 on c_3 to cover both sectors H and K.
6) Since only d_1 remains in $\hat{\mathbf{D}}_\mathbf{L}$ (for Z objects), we finally deploy a sensor s_6 on c_1 to cover sectors B, C, and D.

In this example, we totally place six R&D sensors. Besides, their rotation schedules are presented as follows:

$$s_1 : \{(F, T)\}, s_2 : \{(A, T)\}, s_3 : \{(I, T/2), (J, T/2)\},$$
$$s_4 : \{(E, T/2), (G, T/2)\}, s_5 : \{(H, T/2), (K, T/2)\},$$
$$s_6 : \{(B, T/3), (C, T/3), (D, T/3)\}.$$

We then analyze the time complexity of step 3 in Lemma 3.

Lemma 3. Placing R&D sensors in step 3 spends time of $O(m \lg m)$ in the worst case.

Proof: To iteratively select one disk to be placed with an $R\&D$ sensor, we can construct a maximum binary heap to maintain all disks in $\hat{\mathbf{D}}$. Each disk $d_j \in \hat{\mathbf{D}}$ is sorted (in the heap) based on a two-tuple value (δ_j^{max}, u_j), where δ_j^{max} is the largest δ_i value in d_j, and u_j is the maximum number of uncovered objects in d_j that can be covered by one R&D sensor. A disk d_i is 'larger' than another disk d_j if 1) $\delta_i^{max} > \delta_j^{max}$ or 2) $\delta_i^{max} = \delta_j^{max}$ and $u_i > u_j$. Recall that the set $\hat{\mathbf{D}}$ contains no more than $O(m)$ disks. Therefore, it takes $O(m)$ time to construct the heap. In addition, the time to extract the maximum value (i.e., a disk) from the heap will be $O(\lg m)$. Since there are totally m objects in $\hat{\mathbf{O}}$, we will execute no more than $O(m)$ times of the above extracting operations. Therefore, running step 3 takes at most $O(m) + O(m) \cdot O(\lg m) = O(m \lg m)$ time. □

4.4 Step 4: Removing Redundancy

The aforementioned step 3 places R&D sensors based on the assumption that each sensor can rotate to cover at most $\lfloor 1/\delta_i^{max} \rfloor$ sectors, where δ_i^{max} is the maximum δ_i value of objects that the sensor covers. However, this assumption forces the sensor to waste at least $(1 - \delta_i^{max} \cdot \lfloor 1/\delta_i^{max} \rfloor)T$ monitoring time when $1/\delta_i^{max} \neq \lfloor 1/\delta_i^{max} \rfloor$. For example, supposing that $\delta_i^{max} = 0.55$, then the sensor has to waste $(1 - 0.55 \cdot \lfloor 1/0.55 \rfloor)T = 0.45T$ monitoring time in every period. In fact, this assumption restricts each R&D sensor to covering no more than one sector when $\delta_i^{max} > 0.5$, making its rotatable ability become useless.[2] Therefore, the objective of step 4 is to relax the above assumption, and exploit the wasting time of sensors to further save the number of sensors used in each disk.

In step 4, we check only those disks placed with two or more R&D sensors, and remove redundant sensors from them. Our discussion will focus on *one* such disk, say, d_a. (Other

2. We will also evaluate the relationship between δ_i^{max} and such wasting time in Section 5.3.

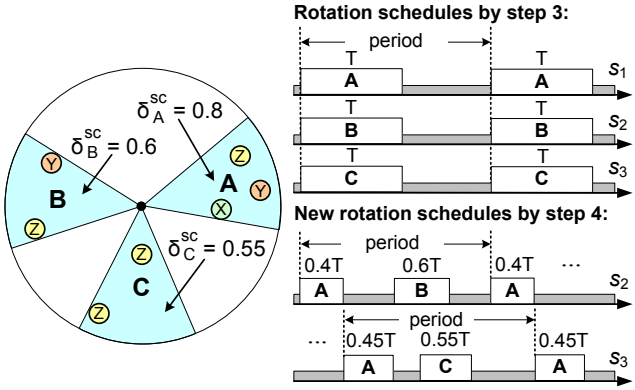


Fig. 7: An example of step 4 in the GRSD heuristic, where X, Y, and Z objects should be 0.8-time, 0.6-time, and 0.55-time covered, respectively.

disks will be handled in the same way.) Suppose that the ASF operation has found a set of sectors $\hat{\mathbf{S}}_\mathbf{a}$ in d_a. We use the notation ρ_j to denote whether a sector $sc_j \in \hat{\mathbf{S}}_\mathbf{a}$ is covered by a sensor s_i, where $\rho_j = 1$ if sc_j is covered by s_i, and $\rho_j = 0$ otherwise. Also, let us denote by δ^{sc}_j the maximum δ_i value of objects in sc_j. Then, the *occupied monitoring time* of s_i, which is the minimum time that s_i has to spend to monitor all of its objects in d_a during a period, is defined by

$$T^{\mathbf{o}}_i = \sum\nolimits_{sc_j \in \hat{\mathbf{S}}_\mathbf{a}} \rho_j \cdot \delta^{sc}_j \cdot T. \quad (2)$$

Therefore, the *residual monitoring time* of s_i can be easily derived by

$$T^{\mathbf{r}}_i = T - T^{\mathbf{o}}_i.$$

Then, we say that a sensor s_i is *redundant* if

$$T^{\mathbf{o}}_i \leq \sum\{T^{\mathbf{r}}_k \mid \text{for each sensor } s_k \neq s_i \text{ that covers } \hat{\mathbf{S}}_\mathbf{a}\}. \quad (3)$$

In this case, for each sector $sc_j \in \hat{\mathbf{S}}_\mathbf{a}$ covered by s_i, we can 'combine' the residual monitoring time of some other sensors in d_a (excluding s_i), so that the length of such combined time is at least $T^{\mathbf{o}}_i$. In this way, s_i can be removed, because all of its sectors have been covered by other sensors.

Fig. 7 presents an example to demonstrate the benefit of step 4. Suppose that the disk has three sectors and contains X, Y, and Z objects, where $\delta_X = 0.8$, $\delta_Y = 0.6$, and $\delta_Z = 0.55$, respectively. According to step 3, three sensors s_1, s_2, and s_3 should be placed on the disk to cover sectors A, B, and C, respectively. In this case, their rotation schedules must be

$$s_1 : \{(A, T)\}, s_2 : \{(B, T)\}, s_3 : \{(C, T)\}.$$

Obviously, the rotatable ability of R&D sensors is not well utilized, because every sensor is allowed to cover at most one sector in each period. In fact, because

$$\begin{aligned} T^{\mathbf{o}}_1 &= 0.8T \\ &< T^{\mathbf{r}}_2 + T^{\mathbf{r}}_3 = (T - 0.6T) + (T - 0.55T) = 0.85T, \end{aligned}$$

sensor s_1 is thus redundant. Therefore, according to step 4, sensors s_2 and s_3 can respectively spend $0.4T$ and $0.45T$ time to cooperatively cover sector A. In this case, s_1 can be removed, and the rotation schedules of other sensors are updated by

$$s_2 : \{(A, 0.4T), (B, 0.6T)\}, s_3 : \{(A, 0.45T), (B, 0.55T)\},$$

which takes advantage of sensor rotation to save the number of R&D sensors required in the disk. It is noteworthy that two sensors cannot *simultaneously* cover the same sector (in other words, their monitoring time for the same sector cannot have any overlap). This is to guarantee that the length of combined residual monitoring time is sufficient to satisfy the coverage demands of all objects in that sector. To do so, we should avoid *aligning* the period boundary of each sensor. Specifically, suppose that two sensors s_i and s_j cover the same sector sc_k. Then, we can allow s_i to start its period by monitoring sc_k first. After s_i finishes monitoring sc_k, s_j can start its period by monitoring sc_k. In this way, we can prevent both s_i and s_j from simultaneously monitoring sc_k. For example, Fig. 7 gives the new rotation schedules of s_2 and s_3 according to step 4. Lemma 4 presents the computation cost of step 4.

Lemma 4. Assume that the maximum number of objects in each disk is $\alpha > 1$. Then, it takes no more than $O(\alpha m)$ time to calculate all redundant R&D sensors.

Proof: Step 4 checks only those disks placed with multiple sensors. The worst case occurs when we have to check all disks in $\hat{\mathbf{D}}$. Therefore, we assume that every disk contains $\alpha > 1$ objects, so $\hat{\mathbf{D}}$ will have m/α disks. For every disk, the ASF operation cuts it into α sectors (i.e., each sector contains one object). By step 3, we place at most α R&D sensors in the disk, where each sensor covers one sector. In this way, it takes $O(\alpha \cdot \alpha)$ time to calculate the occupied monitoring time of all sensors in the disk by Eq. (2), because each sensor has to examine all of the α sectors in the disk. According to Eq. (3), each sensor will compute the sum of the occupied monitoring time of other $(\alpha - 1)$ sensors, in order to check whether the sensor is redundant or not. Thus, the time to find redundant sensors in one disk will be $O(\alpha^2 + \alpha(\alpha - 1))$. Because there are m/α disks in $\hat{\mathbf{D}}$, the overall time complexity to calculate all redundant sensors will be

$$\frac{m}{\alpha} \cdot O(\alpha^2 + \alpha(\alpha - 1)) = O(\alpha m).$$

□

4.5 Step 5: Connecting Nodes

In the above steps, we only deploy R&D sensors to cover all objects in $\hat{\mathbf{O}}$, but they may not necessarily form a connected network. Therefore, we have to add additional *relay nodes* to guarantee the network connectivity, where a relay node is the (stand-alone) communication module of a sensor[3]. Here, we borrow the idea from [28] to place relay nodes. In particular, we construct a minimum spanning tree to connect all R&D sensors. Then, for every tree edge whose length, say, q_i is longer than the communication distance r_c, we add $(\lceil q_i/r_c \rceil - 1)$ relay nodes on that edge, where any two adjacent relay nodes are separated by a distance of r_c. In this way, we can connect the two R&D sensors located on the both end-points of the tree edge by using the minimum number of relay nodes. We remark on the advantages of using relay nodes. First, since a relay node is cheaper than an R&D sensor, the network deployment cost can be reduced. Second, our heuristic allows an arbitrary relationship between r_s and r_c, because it deals with the coverage and connectivity problems separately. Lemma 5 gives the computation cost of step 5, and Theorem 3 presents the total computation complexity of our GRSD heuristic.

Lemma 5. Adding relay nodes by step 5 takes time of $O(m^2)$.

3. This is feasible since many sensor platforms such as MICAz Mote [33] allow a sensor to be separated into *sensing* and *communication* modules.

Proof: Because $|\hat{\mathbf{O}}| = m$, we will deploy at most $O(m)$ R&D sensors to cover all objects. To construct a minimum spanning tree to connect all R&D sensors, we can employ Prim's algorithm. It has been shown in [34] that the computation cost of Prim's algorithm is $O(|\mathbf{E}| + |\mathbf{V}| \cdot \lg |\mathbf{V}|)$ by using Fibonacci heaps, where $\mathbf{V}$ denotes the vertex set while $\mathbf{E}$ represents the edge set. By viewing each R&D sensor as a vertex in $\mathbf{V}$, we have $|\mathbf{V}| = O(m)$. Besides, there are at most $m(m-1)/2$ edges by connecting any pair of sensors, so we have $|\mathbf{E}| = O(m(m-1)/2) = O(m^2)$. Therefore, it spends

$$O(|\mathbf{E}| + |\mathbf{V}| \cdot \lg |\mathbf{V}|) = O(m^2 + m \lg m) = O(m^2)$$

time to construct the minimum spanning tree. In addition, the number of edges in the minimum spanning tree will be $|\mathbf{V}| - 1 = m - 1$. Therefore, it means that we have to check at most $(m-1)$ tree edges (and place relay nodes if necessary). In sum, the computation cost to add relay nodes in step 5 will be

$$O(m^2) + O(m-1) = O(m^2).$$

□

Theorem 3. Given m objects to be covered, the GRSD heuristic has the computation complexity of $O(m^2)$.

Proof: According to Lemmas 1 $\sim$ 5, we can derive the computation complexity of GRSD by

$$O(m^2) + O(2m) + O(m \lg m) + O(\alpha m) + O(m^2).$$

Since α is the maximum number of objects in a disk, we have $\alpha < m$. Therefore, the above equation can be simplified to $O(m^2) + O(\alpha m) = O(m^2)$. □

5 Simulation Study

We develop a Java-based simulator to measure the performance of our GRSD heuristic in terms of the number of nodes deployed. It simulates a square-shaped sensing field whose length is 400. There are three types of static objects, namely X, Y, and Z objects, which need to be δ_X-time, δ_Y-time, and δ_Z-time covered, respectively. Two δ-settings are applied to the simulator as follows:

- **δ-setting 1:** $\delta_X = 0.6$, $\delta_Y = 0.5$, and $\delta_Z = 0.3$.
- **δ-setting 2:** $\delta_X = 0.6$, $\delta_Y = 0.3$, and $\delta_Z = 0.25$.

In both δ-settings, we let δ_X be 0.6, so an R&D sensor can cover only one sector that contains X objects. In this case, most existing methods (discussed in Section 2.2) will force the R&D sensor to stop rotating, and thus the sensor has to give up $1 - \delta_X = 0.4$ portion of its monitoring time. We use this δ_X value to demonstrate the benefit of exploiting the residual monitoring time of R&D sensors (to cooperatively cover objects) in our GRSD heuristic. Furthermore, we use lower δ_Y and δ_Z values in δ-setting 2 to observe whether smaller δ_i values of other objects (in the existence of X objects) can help significantly reduce the number of deployed sensors. In particular, an R&D sensor can rotate to cover one more sector containing only Y or Z objects in δ-setting 2 (compared with that in δ-setting 1).

We also consider two object-placement scenarios to determine the positions of objects in the sensing field:

- **Even object-placement (EOP) scenario:** The number of X, Y, and Z objects are equal. All objects are randomly placed in the sensing field by using the uniform distribution.
- **Uneven object-placement (UOP) scenario:** The percentages of X, Y, and Z objects are 25%, 25%, and 50%, respectively. The sensing field is cut into two equal halves. X objects only appear in the left part, while Y objects only appear in the right part. Z objects are randomly placed in the whole sensing field by using the uniform distribution.

For each R&D sensor, we set $r_s = 15$ and $r_c = 30$. The number of objects and the sector angle θ are varied in the simulator to measure their effect.

We compare our GRSD heuristic with the MCD and DOD methods in [10], whose goal is to deploy the minimum number of R&D sensors to make every object be δ-time covered. Both MCD and DOD compute a set of disks to cover all objects. Then, MCD iteratively deploys an R&D sensor in the disk that contains the maximum number of objects, until all objects are δ-time covered. On the other hand, DOD seeks to deploy sensors to cover the joint sectors of overlapped disks. However, MCD and DOD are designed for *homogeneous* objects. Therefore, we also measure the performance of our GRSD heuristic *without the improvement of step 4* for comparison. We call this method *GRSD-FT*, where 'FT' means 'fixed (monitoring) time'. Without step 4, each R&D sensor can rotate to cover only $\lfloor 1/\delta_i^{max} \rfloor$ sectors. Each sector has the monitoring time of $\delta_i^{max} \cdot T$, where δ_i^{max} is the largest δ_i value of objects that the sensor covers. Step 4 relaxes this assumption, and thus allows sensors to exploit its residual monitoring time to cooperatively cover some sectors. By comparing GRSD with GRSD-FT, we can estimate the effect of step 4. In addition, we define the *node saving ratio* $f(x)$ of GRSD to an x-method by:

$$\frac{(\text{node \# by } x\text{-method}) - (\text{node \# by GRSD})}{\text{node \# by } x\text{-method}} \times 100\%,$$

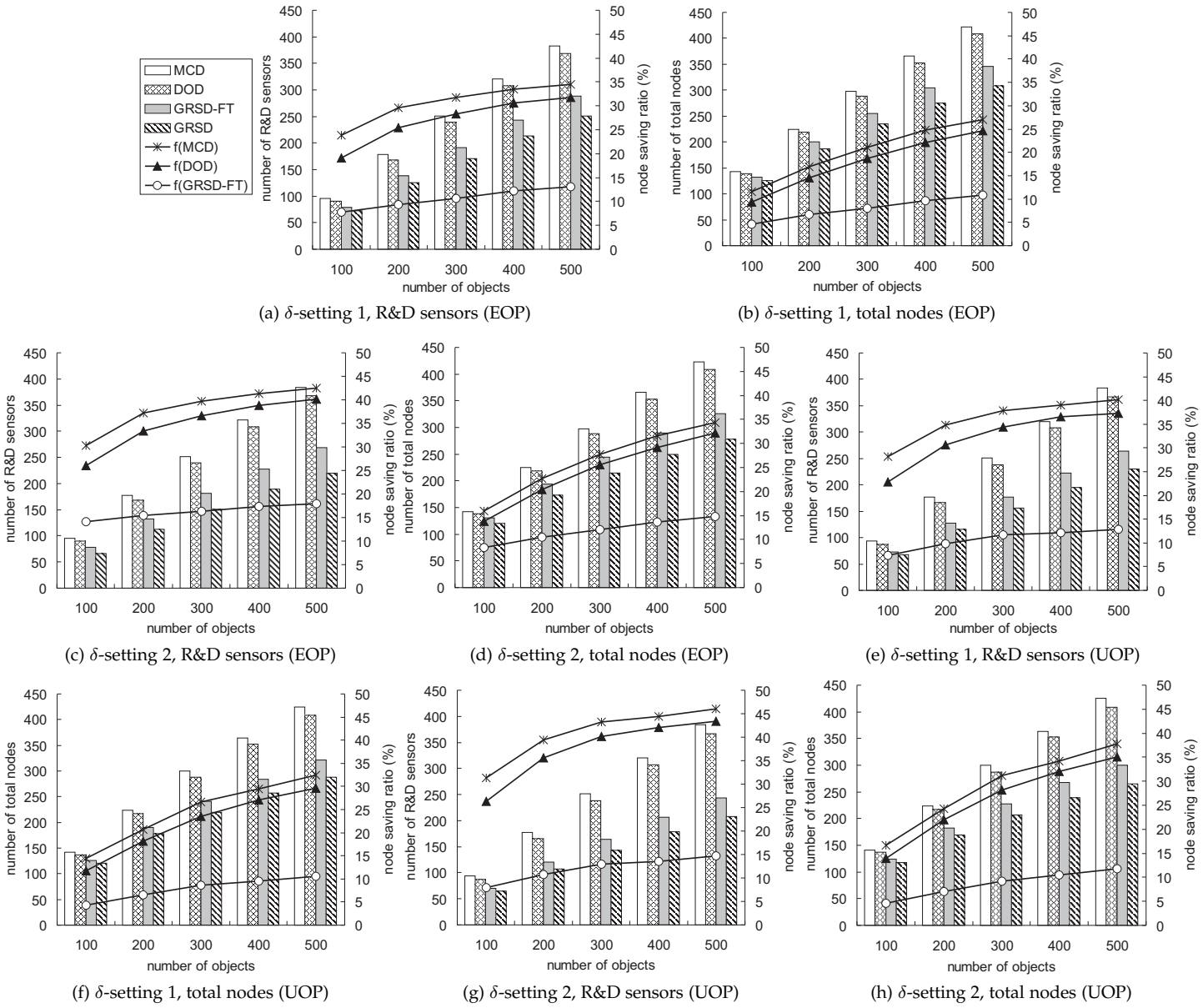
where 'node #' means the number of deployed nodes, and the x-method can be MCD, DOD, or GRSD-FT. This ratio helps us evaluate the performance improvement by GRSD.

5.1 Effect of The Number of Objects

We first measure the effect of different number of objects on the number of nodes deployed by MCD, DOD, GRSD-FT, and GRSD. The number of objects is ranged from 100 to 500. We set the sector angle θ to 30 degrees, and evaluate 1) the number of sensors required to cover all objects and 2) the total number of nodes (including sensors and relay nodes) used to construct the whole network.

Fig. 8(a)–(d) show the number of nodes deployed by MCD, DOD, GRSD-FT, and GRSD by changing the number of objects in the EOP scenario. Apparently, all methods require more R&D sensors when there are more objects. In this case, one may require more relay nodes to connect these sensors. Since MCD and DOD consider homogeneous objects with the same δ value, they have to operate based on δ_X, which is the largest δ_i value of all objects. However, δ_X is set to 0.6 in δ-settings 1 and 2. This forces an R&D sensor to cover at most $\lfloor 1/0.6 \rfloor = 1$ sector in MCD and DOD. That is why MCD and DOD deploy the equal number of nodes in both δ-settings. In this case, DOD works better than MCD, because it can save more sensors by taking advantage of disk overlap.

On the contrary, GRSD-FT and GRSD break the above δ_X restriction, and allow R&D sensors to cover different number of sectors according to their covered objects. This property significantly reduces the number of sensors used to



(a) δ-setting 1, R&D sensors (EOP)

(b) δ-setting 1, total nodes (EOP)

(c) δ-setting 2, R&D sensors (EOP)

(d) δ-setting 2, total nodes (EOP)

(e) δ-setting 1, R&D sensors (UOP)

(f) δ-setting 1, total nodes (UOP)

(g) δ-setting 2, R&D sensors (UOP)

(h) δ-setting 2, total nodes (UOP)

Fig. 8: Comparison on the number of nodes deployed by MCD, DOD, GRSD-FT, and GRSD by changing the number of objects: (a)–(d) in the EOP scenario and (e)–(h) in the UOP scenario.

meet the δ_i-covered requirement of objects. With the help of step 4, GRSD can combine the residual monitoring time of multiple sensors to cover the same sector, thereby removing unnecessary sensors. Therefore, our GRSD heuristic always requires the minimum number of sensors to cover all objects, as compared with other three methods. It can be observed that both GRSD-FT and GRSD save more nodes in δ-setting 2 (than δ-setting 1), because sensors are able to cover more sectors with Y and Z objects. For GRSD-FT, each sensor covers at most $\lfloor 1/0.5 \rfloor = 2$ and $\lfloor 1/0.3 \rfloor = 3$ sectors containing Y and Z objects in δ-setting 1, but it can cover $\lfloor 1/0.3 \rfloor = 3$ and $\lfloor 1/0.25 \rfloor = 4$ sectors containing Y and Z objects in δ-setting 2, respectively. For GRSD, smaller δ_Y and δ_Z values in δ-setting 2 mean that sensors can remain more residual monitoring time, which has a larger opportunity to allow multiple sensors to cooperatively cover the same sector.

On the average, in δ-setting 1, GRSD saves 30.65%, 27.06%, and 10.58% of R&D sensors, and 20.27%, 17.86%, and 7.98% of all nodes compared with MCD, DOD, and GRSD-FT, respectively. In δ-setting 2, GRSD saves 38.18%, 34.97%, and 16.22% of R&D sensors, and 26.45%, 24.23%, and 11.88% of all nodes compared with MCD, DOD, and GRSD-FT, respectively. Notice that GRSD/GRSD-FT require more relay nodes than MCD and DOD.[4] The reason is that GRSD/GRSD-FT deploy fewer R&D sensors than MCD and DOD, and these sensors may have distances more than r_c. In this case, more relay nodes are used to connect all sensors. Explicitly, a larger r_c value can help reduce the number of relay nodes.

Fig. 8(e)–(h) present the number of nodes deployed by MCD, DOD, GRSD-FT, and GRSD by changing the number of objects in the UOP scenario. When there are more objects, each method requires more nodes to construct the network. Both MCD and DOD deploy the similar number of nodes in the UOP scenario compared with that in the EOP scenario, because they highly depend on the δ_X value. GRSD-FT and GRSD, on the other hand, save more R&D sensors in the UOP scenario

4. GRSD and GRSD-FT require the equal number of relay nodes since step 4 of GRSD only removes redundant sensors in the same disk.

due to two reasons. First, there are around 33.3% and 50% of Z objects (which has the smallest δ_i value) in the EOP and UOP scenarios, respectively. This allows sensors to cover more sectors in the UOP scenario. Second, X objects only appear in the left part of the sensing field in the UOP scenario. In this case, GRSD-FT and GRSD can use fewer sensors to cover the objects in the right part of the sensing field.

On the average, in δ-setting 1, GRSD saves 36.02%, 32.37%, and 10.78% of R&D sensors, and 24.75%, 22.03%, and 7.90% of all nodes compared with MCD, DOD, and GRSD-FT, respectively. In δ-setting 2, GRSD saves 40.89%, 37.50%, and 11.97% of R&D sensors, and 28.80%, 26.23%, and 8.61% of all nodes compared with MCD, DOD, and GRSD-FT, respectively. This verifies the effectiveness of GRSD in the UOP scenario.

From Fig. 8, we can have the following observations. First, increasing the number of objects has great impact on the number of nodes used to construct the network. Our GRSD heuristic always requires the minimum number of sensors to cover all objects. Second, the performance of MCD and DOD are dominated by the largest δ_i value (i.e., δ_X), so changing other δ_i values and the placement of objects may have less effect (when δ_X does not change). Third, using smaller δ_Y and δ_Z values or altering the object placement can significantly improve the performance of both GRSD-FT and GRSD.

5.2 Effect of The Sector Angle

We then evaluate the effect of different sector angles θ on the number of R&D sensors deployed by MCD, DOD, GRSD-FT, and GRSD. There are 200 and 400 objects placed in the sensing field. Beginning from 120 degrees, θ is gradually decreased by 15 degrees, until it reaches 15 degrees. Since the θ value decides the number of sectors in each disk, but it does not affect the number of relay nodes, we only estimate the number of R&D sensors in this experiment.

Fig. 9 presents the number of R&D sensors deployed by MCD, DOD, GRSD-FT, and GRSD under different sector angles θ. Obviously, a smaller θ value means that a sensor covers a narrower range, so fewer objects can be covered. In this case, all methods require more sensors as θ decreases. Generally speaking, we have MCD $>$ DOD $>$ GRSD-FT $>$ GRSD in terms of the number of sensors, which demonstrates that GRSD is an outstanding deployment scheme compared with other three methods. In addition, we have the following two observations from Fig. 9:

1) The node saving ratios $f(\text{MCD})$ and $f(\text{DOD})$ significantly grow when the θ value decreases. The reason is that both MCD and DOD determine the number of sectors covered by a sensor according to δ_X. In this experiment, we have $\delta_X = 0.6$, so each sensor is allowed to cover at most one sector. In this case, a smaller sector angle θ implies that each disk could be cut into more sectors, thereby making MCD and DOD deploy more sensors to cover objects.
2) Changing the θ value has slight effect on the node saving ratio $f(\text{GRSD-FT})$. In the EOP scenario, $f(\text{GRSD-FT})$ grows when θ diminishes in the δ-setting 1 (referring to Fig. 9(a) and (c)). However, the ratio decreases as θ decreases in the δ-setting 2 (referring to Fig. 9(b) and (d)). The reason to cause such phenomenon is that GRSD-FT allows sensors to cover more sectors containing only Y and Z objects in δ-setting 2. On the other hand, $f(\text{GRSD-FT})$ grows as θ reduces in the UOP scenario (referring to Fig. 9(e)–(h)). In this scenario, there is a higher opportunity that GRSD can find more redundant sensors in the right part of the sensing field (because this part contains only Y and Z objects), and remove them accordingly.

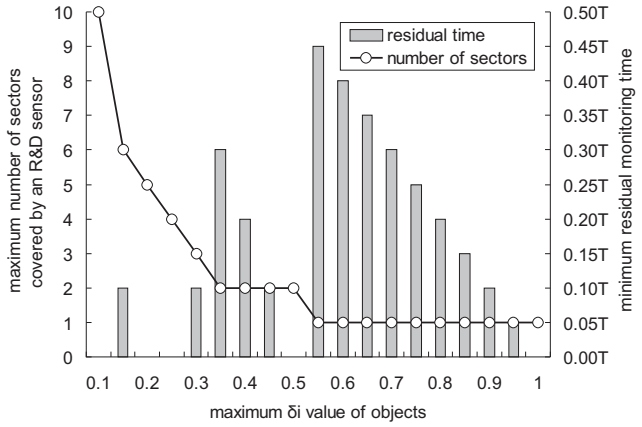


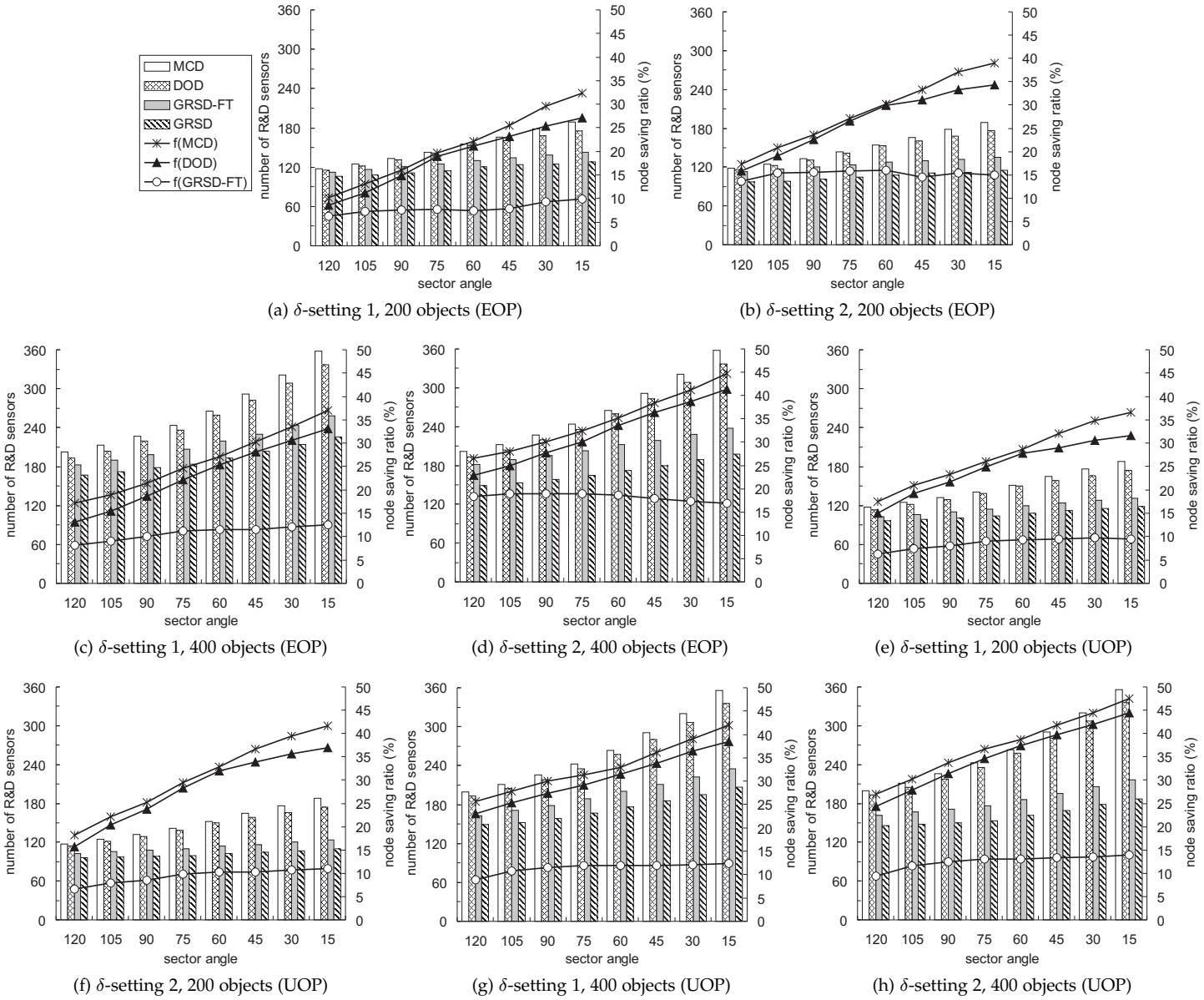
Fig. 10: Effect of different δ_i values on the maximum number of sectors covered by an R&D sensor and its minimum residual monitoring time.

5.3 Effect of δ_i Values

In MCD, DOD, and GRSD-FT, the maximum number of sectors covered by one R&D sensor depends on the δ_i^{max} value (i.e., the maximum δ_i value of objects). Besides, the sensor may remain some residual monitoring time when 1) the sensor covers fewer than $\lfloor 1/\delta_i^{max} \rfloor$ sectors or 2) $\lfloor 1/\delta_i^{max} \rfloor \neq 1/\delta_i^{max}$. Fig. 10 shows the effect of different δ_i^{max} values on the maximum number of sectors covered by an R&D sensor and its minimum residual monitoring time (when the sensor has covered $\lfloor 1/\delta_i^{max} \rfloor$ sectors). In this experiment, starting from $\delta_i^{max} = 0.1$, we gradually increase δ_i^{max} by 0.05, until it reaches 1. From Fig. 10, we can observe that the maximum number of sectors covered by a sensor drastically decreases as δ_i^{max} grows, especially when $\delta_i^{max} < 0.35$. When $\delta_i^{max} > 0.5$, each R&D sensor can cover only one sector. In this case, R&D sensors become *static*, so the performance of MCD, DOD, and GRSD-FT significantly decreases. Furthermore, except for certain δ_i^{max} values (specifically, 0.1, 0.2, 0.25, 0.5, and 1), R&D sensors must have positive residual monitoring time. When $\delta_i^{max} = 0.55$, a sensor even wastes around half of its monitoring time (i.e., the residual monitoring time is $0.45T$) in every period. That is why we develop step 4 in our GRSD heuristic to well utilize such residual monitoring time. By combining the residual monitoring time of two or more R&D sensors in the same disk, GRSD can check for redundant sensors, and remove them to save the overall deployment cost.

5.4 Effect of Sparse Objects

The experiments in both Section 5.1 and Section 5.2 consider a large-scale scenario, where there are 100 $\sim$ 500 objects in the sensing field. In this case, some disks in $\hat{\mathbf{D}}$ may contain a large number of objects in $\hat{\mathbf{O}}$, so different deployment schemes can employ fewer R&D sensors to cover the objects in these disks. In contrast with the previous experiments, we evaluate the effect of *sparse objects* on the number of R&D sensors deployed in this section. Specifically, the sensing field contains only 50 objects. These objects are sparsely distributed over the sensing field, such that each disk covers 2 $\sim$ 4 objects. In addition,



(a) δ-setting 1, 200 objects (EOP)

(b) δ-setting 2, 200 objects (EOP)

(c) δ-setting 1, 400 objects (EOP)

(d) δ-setting 2, 400 objects (EOP)

(e) δ-setting 1, 200 objects (UOP)

(f) δ-setting 2, 200 objects (UOP)

(g) δ-setting 1, 400 objects (UOP)

(h) δ-setting 2, 400 objects (UOP)

Fig. 9: Comparison on the number of R&D sensors deployed by MCD, DOD, GRSD-FT, and GRSD by changing the sector angle θ: (a)–(d) in the EOP scenario and (e)–(h) in the UOP scenario.

every object in $\hat{\mathbf{O}}$ appears in only one disk, which means that disks have no overlap with each other.

Fig. 11 presents the experimental result. Because there is no disk overlap, DOD will work similarly with MCD. Thus, both methods deploy the similar number of sensors to cover all objects. Besides, different δ settings have almost no effect on MCD and DOD, because their performance depends on δ_X (whose value is 0.6 in both δ settings). However, in the UOP scenario, there is a higher possibility that the two methods can use fewer sensors to cover Y and Z objects in the right part of the sensing field. Therefore, MCD and DOD can work better in the UOP scenario, as compared with the EOP scenario. On the other hand, by considering object heterogeneity, GRSD-FT and GRSD require fewer R&D sensors to achieve the temporal coverage of objects. Our GRSD heuristic can always have the best performance even though the objects are sparely located in the sensing field. The reason is that our heuristic can adaptively adjust the rotation schedules of sensors by exploiting their residual monitoring time. From Fig. 11, we can observe that our GRSD heuristic can further save the network deployment cost in the UOP scenario and δ-setting 2. This observation is the same with that in Section 5.1.

6 Conclusion

Given a set of heterogeneous objects, this paper formulates a GRSD problem to determine the positions and rotation schedules of R&D sensors, in order to provide temporal coverage for these objects. The GRSD problem is proven to be NP-hard, so we develop an efficient heuristic whose objective is to save the network deployment cost. Our GRSD heuristic finds a set of disks to cover all objects, and deploys R&D sensors based on these disks to meet the δ_i-time covered demand of each object. Then, it combines the residual monitoring time of R&D sensors in the same disk to make them cooperatively monitor objects. Thus, redundant sensors can be further removed. Simulation results show that the GRSD heuristic significantly reduces the number of R&D sensors under different scenarios, as compared with the MCD, DOD, and GRSD-FT methods. This demonstrates the effectiveness of the proposed GRSD heuristic in terms of the network deployment cost.

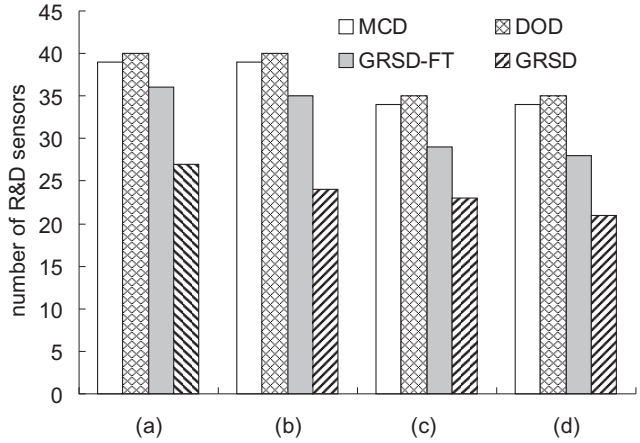


Fig. 11: Comparison on the number of R&D sensors deployed by MCD, DOD, GRSD-FT, and GRSD when objects are sparsely located in the sensing field: (a) δ-setting 1 in the EOP scenario, (b) δ-setting 2 in the EOP scenario, (c) δ-setting 1 in the UOP scenario, and (d) δ-setting 2 in the UOP scenario.

REFERENCES

[1] 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.

[2] Y.C. Wang, "Mobile sensor networks: system hardware and dispatch software," *ACM Computing Surveys*, vol.47, no. 1, pp. 12:1–12:36, 2014.

[3] M.A. Guvensan and A.G. Yavuz, "On coverage issues in directional sensor networks: a survey," *Ad Hoc Networks*, vol. 9, no. 7, pp. 1238–1255, 2011.

[4] S.A.L. Glegg, M.P. Olivieri, R.K. Coulson, and S. M. Smith, "A passive sonar system based on an autonomous underwater vehicle," *IEEE J. Oceanic Engineering*, vol. 26, no. 4, pp. 700–710, 2001.

[5] G. Lee and N.Y. Chong, "Low-cost dual rotating infrared sensor for mobile robot swarm applications," *IEEE Trans. Industrial Informatics*, vol. 7, no. 2, pp. 277–286, 2011.

[6] 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.

[7] 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.

[8] Y. Zhang and S. Wang, "Remote relative wind velocity estimation using airborne Doppler radar and spectrum analysis," *IEEE Trans. Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 1648–1667, 2011.

[9] N.I. Giannoccaro, L. Spedicato, and C. di Castri, "A new strategy for spatial reconstruction of orthogonal planes using a rotating array of ultrasonic sensors," *IEEE Sensors J.*, vol. 12, no. 5, pp. 1307–1316, 2012.

[10] 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.

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

[12] 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*, 2005, pp. 246–253.

[13] C. Liu and G. Cao, "Spatial-temporal coverage optimization in wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 4, pp. 465–478, 2011.

[14] 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.

[15] M.R. Senouci, A. Mellouk, and K. Assnoune, "Localized movement-assisted sensor deployment algorithm for hole detection and healing," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1267–1277, 2014.

[16] 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.

[17] H. Mahboubi, K. Moezzi, A.G. Aghdam, and K. Sayrafian-Pour, "Distributed deployment algorithms for efficient coverage in a network of mobile sensors with nonidentical sensing capabilities," *IEEE Trans. Vehicular Technology*, vol. 63, no. 8, pp. 3998–4016, 2014.

[18] 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.

[19] 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.

[20] N. Bisnik, A.A. Abouzeid, and V. Isler, "Stochastic event sapture using mobile sensors subject to a quality metric," *IEEE Trans. Robotics*, vol. 23, no. 4, pp. 676–692, 2007.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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, 2009.

[27] H. Yang, D. Li, and H. Chen, "Coverage quality based target-oriented scheduling in directional sensor networks," *Proc. IEEE Int'l Conf. Comm.*, pp. 1–5, 2010.

[28] 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.

[29] 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, no. 2, pp. 216–225, 2010.

[30] D. Tao and T.Y. Wu, "A survey on barrier coverage problem in directional sensor networks," *IEEE Sensors J.*, vol. 15, no. 2, pp. 876–885, 2015.

[31] J.Y. Lee, "Exploiting constrained rotation for localization of directional sensor networks," *Proc. IEEE Int'l Wireless Comm. and Mobile Computing Conf.*, 2008, pp. 767–772.

[32] G.K. Das, R. Fraser, A. Lopez-Ortiz, and B.G. Nickerson, "On the discrete unit disk cover problem," *Lecture Notes in Computer Science*, vol. 6552, pp. 146–157, 2011.

[33] N.A. Ali, M. Drieberg, and P. Sebastian, "Deployment of MICAz Mote for wireless sensor network applications," *Proc. IEEE Int'l Conf. Computer Applications and Industrial Electronics*, pp. 303–308, 2011.

[34] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, 2001.