# Static and Mobile Target $k$-Coverage in Wireless Rechargeable Sensor Networks

Pengzhan Zhou, Cong Wang, and Yuanyuan Yang, *Fellow, IEEE*

**Abstract**—Energy remains a major hurdle in running computation-intensive tasks on wireless sensors. Recent efforts have been made to employ a Mobile Charger (MC) to deliver wireless power to sensors, which provides a promising solution to the energy problem. Most of previous works in this area aim at maintaining perpetual network operation at the expense of high operating cost of MC. In the meanwhile, it is observed that due to the low cost of wireless sensors, they are usually deployed at high density so there is abundant redundancy in their coverage in the network. For such networks, it is possible to take advantage of the redundancy to reduce the energy cost. In this paper, we relax the strictness of perpetual operation by allowing some sensors to temporarily run out of energy while still maintaining target $k$-coverage in the network at lower cost of MC. We first establish a theoretical model to analyze the performance improvements under this new strategy. Then, we organize sensors into load-balanced clusters for target monitoring by a distributed algorithm. Next, we propose a charging algorithm named $\lambda$-GTSP Charging Algorithm to determine the optimal number of sensors to be charged in each cluster to maintain $k$-coverage in the network and derive the route for MC to charge them. We further generalize the algorithm to encompass mobile targets as well. Our extensive simulation results demonstrate significant improvements of network scalability and cost saving that MC can extend charging capability over 2-3 times with a reduction of 40 percent of moving cost without sacrificing the network performance.

**Index Terms**—Wireless sensor networks, wireless charging, target $k$-coverage, route planning

## 1 Introduction

Recent advances in technology are leading the trend to launch intelligent applications. These applications usually rely on ubiquitous wireless sensors to capture and generate a huge amount of data from multi-dimensions to detect, recognize and classify objects/targets with high accuracy. However, energy consumption still remains a major challenge for wireless sensors to conduct intensive data processing, computation and communication. As an emerging technology, wireless charging has provided a convenient way to charge the battery of a sensor without wires or plugs. Sensors can be charged in distance by either deploying static wireless transmitters [1], [2] or *Mobile Chargers* (MCs) [4], [5], [7], [8], [11].

Most of the previous works consider perpetual network operation as an ultimate goal whereas such ambition usually comes at high cost. To ensure no sensor ever depletes energy, MC must respond to energy demands all over the network at any time and anywhere. It not only complicates system designs (algorithms), but also makes it difficult to implement in large networks with hundreds of sensors. In addition, these works have not jointly considered charging and balancing the workload of sensors to make charging of MC more efficient.

Low-cost wireless sensors are usually deployed at high density so there is abundant redundancy in their coverage. Taking advantage of such redundancy, we can relax the strictness of perpetual operation by allowing some nodes to temporarily stay in zero energy status while still maintaining the network functionality. To evaluate the sensing quality of such a strategy, we consider a typical task, monitoring a set of *targets* in a *Wireless Sensor Network* (WSN). Rather than keeping *full-coverage* of targets all the time which requires turning on all the sensors, we allow *$k$-coverage* of targets [12], where $k$ is a user-input based on various task requirements. In fact, with advances in machine learning, $k$-coverage should be sufficient for many applications, e.g., in a security monitoring application, images from $k$ camera sensors taken at different angles can recognize objects with high accuracy.

Motivated by these observations, in this paper, we propose a new framework, called $k$-coverage *Wireless Rechargeable Sensor Network* (WRSN), where sensors are organized into clusters around each target and it is required that at least $k$ sensors should be working in each cluster at any moment to engage in sensing tasks to cover the target. In the meanwhile, MC is adopted to meet energy demands from clusters. The MC is usually more expensive compared with sensors, which motivates us to improve its utility. In deriving performance improvements, we focus on charging capability of one MC but the results can be easily scaled to adopt multiple MCs. In particular, the new framework raises several new challenges. First, to what extent does MC improve its charging capability under the new framework? Second, how are sensors organized around targets autonomously and how do clusters balance workload to make wireless charging more efficient? Third, how many sensors should MC charge in each cluster while still satisfying target $k$-coverage and what is the optimal route planning strategy for MC? Finally, what if targets can move? Can we extend the algorithm to handle targets with mobility?

- *P. Zhou and Y. Yang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: {Pengzhan.zhou, yuanyuan.yang}@stonybrook.edu.*
- *C. Wang is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529. E-mail: c1wang@odu.edu.*

*Manuscript received 10 Sept. 2017; revised 5 July 2018; accepted 30 Aug. 2018. Date of publication 28 Sept. 2018; date of current version 28 Aug. 2019. (Corresponding author: Yuanyuan Yang.)*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TMC.2018.2872576*

To answer these questions, we first theoretically examine the potential improvements in charging capability of MC in terms of maximum distance it can cover in a one-dimensional network and extend the result to a two-dimensional network as well. Then we consider organizing sensors around each target into a cluster and develop an iterative and distributed algorithm to assign sensors in overlapped regions to different clusters, and achieve uniform cluster size. To find the optimal number of sensors MC should charge in each cluster, we formulate the problem into an Integer Programming (IP) problem and propose a new charging algorithm called $\lambda$-GTSP Charging Algorithm. Finally, we establish a model to characterize mobile targets as *Brownian Motions* [15] and expand the original clusters to retain $k$-coverage of mobile targets. Realizing that clusters cannot be expanded forever, we further derive a condition to characterize when the process should terminate and a new round of clustering should start.

The contributions of this paper is summarized as follows. First, we propose a new framework that relaxes the stringent full-coverage requirement in a WRSN to $k$-coverage while maintaining network functionality. We theoretically prove the improvement in charging capability of MC under our new framework. To the best of our knowledge, this is the first work that attempts to optimize network cost from the perspectives of target coverage/sensor load balancing. Second, we formulate the charging problem into an optimization problem in which MC is scheduled to only charge a portion of zero-energy nodes in each cluster. The actual number could be dynamic and different for different clusters as long as the network manages to maintain target $k$-coverage. Third, in contrast to most of the previous work which only focuses on static targets, we extend our work to cover mobile targets as well. Finally, we conduct extensive simulations to evaluate performance and compare with the previous work [6], [7], [8]. Our results indicate that the new framework can extend charging capability of MC significantly (over 3 times of covering area) and reduce about 40 percent of operating cost of MC. Meanwhile, more than 80 percent of target coverage rate is maintained for mobile targets during operation.

The rest of the paper is organized as follows. Section 2 studies the related works. Section 3 introduces the network model, assumptions and motivations of our work. Section 4 theoretically compares the charging capability of MC based on different charging strategies. Section 5 studies how to balance the cluster size. Section 6 derives the number of sensors to be charged in each cluster and plans the charging route for MC. Section 7 further considers mobile targets. Section 8 evaluates the performance of the new framework and Section 9 concludes the paper.

## 2 Related Works

Integrating wireless energy transfer to sensor networks has been extensively studied recently. In [1], deployment problems of wireless transmitters are studied to extend network lifetime. In [2], adjusting the power level of wireless transmitters such that overall electromagnetic radiations do not exceed a safety threshold is studied. In [3], a new kind of mobile data gathering mechanism named *SenCar* is proposed to achieve longer network lifetime compared with static observer or other mobile observers. Inductive wireless charging is studied in [4], [5], [6], [7], [8], [11]. Since this technique is able to deliver hundreds watts of energy over short distance, MC is usually employed to approach sensors in close proximity for high charging efficiency. In [4], resonant repeaters are utilized in a new scheme to more effectively respond to dynamic energy demands and cover more nodes in WRSN. In [5], a hybrid network consisting of nodes with different energy sources is proposed to reduce the energy consumption of the network while maintaining the performance. In [6], a greedy algorithm is designed to find a charging sequence to maximize network lifetime. In [7], the shortest Hamilton cycle is pre-planned through all the sensors for wireless charging. In [8], MC receives real-time energy status from sensors and makes charging decisions on-the-fly. In [11], joint wireless charging and mobile data gathering is considered.

The problem of $k$-coverage has been studied for WSNs. In [13], Li et al. utilized $k$-order Voronoi diagram to achieve $k$-coverage sensor deployment in a localized manner. In [14], the $k$-area coverage algorithm is extended to the $k$-surface coverage for 3-D surface. These work provides good guidance, but we cannot implement their schemes directly due to: 1) triangular inequality does not hold anymore on irregular 3D surface; the fundamental Traveling Salesmen Problems becomes difficult to solve; 2) clustering and load balancing algorithms need to be re-designed since the shape of clusters changes regarding to the terrain dynamics. Thus, in this paper, we focus on solutions in the 2D space.

TABLE 1
List of Important Notations

| Notation | Definition |
|---|---|
| $k$ | Required number of working sensors in each cluster |
| $\mathcal{N}$ | Set of sensors |
| $n$ | Number of sensors |
| $m$ | Number of targets in the field |
| $r_s$ | Sensing range of a sensor node |
| $\eta$ | Charging threshold of a cluster |
| $T_i, C_i, H_i$ | Target, cluster and cluster head of cluster $(i)$, respectively |
| $n_i$ | Number of nodes in cluster $i$ |
| $\mathcal{B}$ | Set of sub-clusters. |
| $K$ | Number of sub-clusters in each cluster. |
| $l$ | Estimated lifetime of a sensor with full battery |
| $l_i$ | Estimated lifetime of cluster $i$ |
| $L_i$ | Maximum lifetime of cluster $i$ |
| $v$ | Moving speed of MC |
| $\Delta t$ | Time to charge a zero energy sensor to full capacity |
| $\lambda_i$ | Number of sensors to be charged in cluster $i$ |
| $\mu$ | Drift velocity of the mobile target. |
| $t_{re}$ | Time for re-clustering for mobile targets |
| $R$ | Area traversed by the dynamic cluster before re-clustering. |

## 3 Preliminaries

In this section, we describe the network model and assumptions of our work. Important notations used in this paper are summarized in Table 1.

### 3.1 Network Model and Assumptions

We assume that a set of sensors, $\mathcal{N}$, are uniformly randomly distributed in a sensing field, $\mathcal{A}$, to monitor a set of identical

targets, $\mathcal{T}$. The number of sensors and targets in the field are $n$ and $m$ respectively. In addition, there is a Mobile Charger (MC) equipped with a high-capacity battery, moving around in the field to charge sensors wirelessly. When a sensor depletes its energy, MC can deliver power to the sensor in proximity. When MC depletes its own energy, it returns to the base station to replace its battery.

In this paper, we adopt the magnetic coupling based wireless charging [21]. This scheme allows effective flow of large amount of energy over a short distance. We do not adopt radiation-based strategies that could charge multiple sensors because of the restriction on the wireless energy emitted. It is often limited to 4W by FCC [22] and exceeding such limit could cause health hazard. In our model, since charging efficiency declines rapidly with the increment of distance, the MC should approach sensors in close proximity for effective charge. The effective charging range is about 0.5 meters. With the latest advance in wireless charging technologies, the MC might charge multiple sensors in distance in a foreseeable future. However, since sensing range is much larger than the charging distance, charging multiple sensors could yield low energy efficiency.

We first analyze the case when targets are static [23], [24], [25]. In practice, many applications require sensors to monitor static targets (scene/view) such as security surveillance, traffic monitoring, etc. Then we targets with slow motion. Sensors have two operating modes, namely, working mode and sleeping mode. In sleeping mode, sensors switch off CPU/radio/sensing devices to save energy and we ignore energy consumption in sleeping mode. All the nodes have sensing range $r_s$. To aggregate data/samples of targets, sensors within $r_s$ distance of targets are organized into clusters, $\mathcal{C}$, where $n_i$ sensors in the $i$th cluster denoted as $C_i$, monitor target $T_i$. In particular, if transmission range $r \geq r_s$, data transmission within a cluster can be done in 2-hop communication with minimum overhead.

Since classification error and noise persist in the state-of-the-art sensing devices, data fusion from multiple sensors can improve the sensing quality and decision accuracy [26]. Thus, we require $k$ sensors to stay in *working mode* in each cluster at any time to monitor the target. In this case, we say sensors around a target provide *k-coverage* where $k$ is a user-input depending on application specifics. $k$ can be as small as 1 (1-coverage). On the other hand, $k$ can also be extended to $n_i$ to provide full-coverage, which has been the predominant method in previous works [1], [2], [7], [8]. Clearly, it incurs higher operating cost for MC to satisfy energy demands of all the sensors in full-coverage.

We follow a $k$-coverage sensor scheduling approach. Initially, all the sensors have full battery capacity with lifetime $l$. During operation, exact $k$ sensors are in working mode, while others remain in sleeping mode. Before the first batch of $k$ sensors deplete their energy, they randomly appoint the next batch of $k$ sensors (with full energy) to continue monitoring. If the cluster has less than $\eta$ percentage of alive nodes, the cluster head sends a charging request to MC. We further define the *maximum lifetime* of cluster $i$ to be $L_i = \lfloor \frac{n_i}{k} \rfloor l$, where $\lfloor \frac{n_i}{k} \rfloor$ is the largest integer no greater than $\frac{n_i}{k}$. $L_i$ is the time duration until the cluster can no longer provide $k$-coverage. When a sensor depletes its energy, it is also turned into sleeping mode and waits for MC to charge.

The MC starts from the base station at a speed of $v$ m/s to fulfill energy requests received from sensors. The time to charge the battery of a sensor from empty to full capacity is $\Delta t$. For $k$-coverage, MC charges $\lambda_i$ nodes in cluster $C_i$. We optimize the choice of $\lambda_i$ in Section 6.3. We make additional assumptions as follows: 1) Sensors know their positions by one-time configuration at the beginning; 2) The base station calculates the charging route, and the route is sent to MC through long range wireless communications.

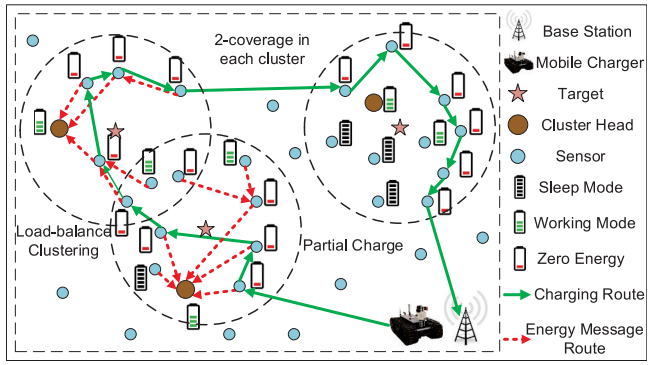



Fig. 1. An example of $\lambda$-GTSP charging framework providing target 2-coverage for three targets.

### 3.2 Generalized Traveling Salesmen Problem

Since we only charge a portion of sensors, $\lambda_i$, in each cluster $C_i$ to keep $k$-coverage, a new route planning approach for MC is needed. Most of the previous works directly adopt the solution for *Traveling Salesmen Problem* (TSP) to establish a Hamiltonian path through all the sensors. That is, the classic TSP requires MC to visit all the nodes with energy charging requests. In our $k$-coverage problem, we explore a generalization of TSP called the *Generalized Traveling Salesmen Problem* (GTSP) [19]. In GTSP, a salesman needs to find the shortest path through some mutually exclusive sets of cities and the path only includes one city from each set. In close analogy, our objective is to find the shortest charging path through clusters of sensors in which MC visits $\lambda_i$ nodes in cluster $C_i$. Hence, we call our new problem $\lambda$-*GTSP*.

The question is whether $\lambda$-GTSP can help us reduce the moving cost of MC. The full-coverage in [7], [8] requires MC to satisfy all energy charging requests. For a rectangular sensing field of side length $D_1$ and $D_2$, a deterministic upper bound of the shortest path traversing $n$ nodes with charging requests is derived as $\sqrt{2(n-2)D_1D_2} + 2(D_1 + D_2)$ [20]. In contrast, in $k$-coverage, we only charge $\lambda_i$ nodes for cluster $C_i$ so $n$ is reduced to $\sum_i \lambda_i$. Since $\sqrt{2(n-2)D_1D_2}$ is much larger than $2(D_1 + D_2)$, the ratio of the tour length of MC derived by $\lambda$-GTSP to the length derived by TSP can be approximated as

$$\frac{\sqrt{2(\sum_i \lambda_i - 2)D_1D_2} + 2(D_1 + D_2)}{\sqrt{2(n-2)D_1D_2} + 2(D_1 + D_2)} \approx \sqrt{\frac{\sum_i \lambda_i}{n}}. \quad (1)$$

We can see that the cost saving to adopt $k$-coverage is proportional to the square root of the ratio of numbers of nodes charged. In practice, the actual number of nodes $\sum_i \lambda_i$ is much smaller than $n$. Since $k$-coverage consumes much less energy than all-coverage, our new approach should reduce moving cost of MC significantly.

Fig. 1 shows an example of the $\lambda$-GTSP charging framework with target 2-coverage for 3 targets. The left two clusters have overlapped regions, in which sensors are assigned

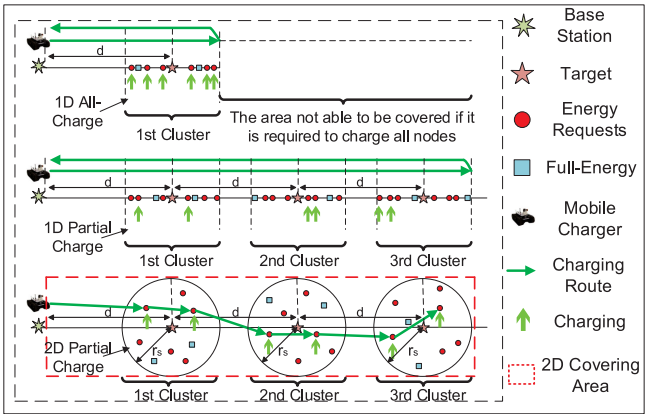



Fig. 2. Comparison of the scope covered by all-charge and partial-charge.

to two clusters evenly for load balance. After clustering, sensors send their energy status to the cluster heads through the red dashed message routes (not drawn in the upper-right cluster for clarity) and the cluster heads send charging requests to the base station if the remaining energy in the cluster is below the threshold. Then MC starts from the base station and charges zero energy sensors following the green charging route. It responds to as many requests as possible while assuring target 2-coverage. Thus, it only charges a portion of sensors in the first cluster so sensors from other clusters can be served on time.

## 4 Charging Capability of MC in a $k$-Coverage Network

Motivated by the potential cost saving of $k$-coverage networks, in this section, we theoretically derive the charging capability of MC, which is represented by the scope or distance of a field MC can cover. We compare it with the solutions in previous works [7], [8], where all energy demands are fulfilled in a single charging round. Since MC is usually much more expensive compared to sensors, it is desirable to extend its covering capability as much as possible. Therefore, for a certain field, fewer MCs are needed in our framework. Our analysis focuses on one MC but the results can be easily scaled for multiple MCs. For analytical tractability, we first conduct the theoretical analysis in a one-dimensional network and then give the conditions on when the results for one-dimensional networks can be applied to two-dimensional networks. We will also examine the performance of two-dimensional networks by simulations in Section 8.

### 4.1 Covering Capability of MC in a One-Dimensional Network

As mentioned in the previous section, MC starts from the base station to fulfill charging requests from $m$ clusters. The sensors within $r_s$ distance of each target are assigned to that target to form a cluster. At any time, in each cluster, $k$ sensors are in working mode while others are in sleeping mode. If the cluster has less than $\eta$ percentage of alive nodes, it sends out a charging request. The lifetime of a sensor is $l$, the charging time from zero to full capacity is $\Delta t$, and the speed of MC is $v$. In addition, for simplicity, we assume that the number of sensors in each cluster is a constant $c$ and the distance between two consecutive targets is a constant $d$ as shown in Fig. 2.

The top two cases in Fig. 2 compare our approach with the previous approach for $k$-coverage in a one-dimensional network. Note that, for fairness, previous approach also only needs to assure target $k$-coverage instead of target full-coverage. The first approach (the previous approach) requires MC to satisfy *all* energy demands whereas the second approach (our approach) only requires to satisfy *partial* energy demands. In particular, for the second approach, we assume that MC charges an equal number of $\lambda$ nodes in each cluster. In fact, $\lambda$ could be different for different clusters based on their energy status and we will further optimize the value of $\lambda$ in Section 6. We compare the covering distance of the two approaches.

#### 4.1.1 All-Charge Approach

It is not difficult to see that MC needs to charge at least $(1-\eta)c$ nodes in each cluster. Before the arrival of MC, each cluster has residual lifetime $l_i \approx \eta cl/k$, where $\eta c$ is the number of sensors that can work. Denote the number of clusters MC can charge before the lifetime of any cluster expires as a variable $x$. The following inequality holds

$$x\left[(1-\eta)c\Delta t + \frac{d}{v}\right] \le \min\left\{\frac{\eta cl}{k}, \frac{(1-\eta)cl}{k}\right\}, \tag{2}$$

where $(1-\eta)c\Delta t$ is the charging time needed in a cluster and $d/v$ is the traveling time of MC between two consecutive clusters. The total time spent over $x$ clusters should be less than the lifetime $l_i$ of each cluster. In addition, it should also be less than the increment of lifetime (due to charging) of a cluster denoted by $(1-\eta)cl/k$ to guarantee perpetual operation. In this way, at the end of a charging round, the ratio of the remaining energy to the full energy in each cluster should be no less than $\eta$ so MC can cover such $x$ clusters in a long run. Here we stipulate that when $(d/v)/((1-\eta)c\Delta t) \le \epsilon$, i.e., the ratio of traveling time to charging time is a very small value $\epsilon$, so the traveling time can be ignored. Note that $\epsilon$ is determined by the accuracy requirement. Even for large networks, traveling time between two targets hundreds of meters apart (1-5 minutes) is still quite small compared to the charging time (60 minutes). This condition for $d$ can be written as

$$d \le \epsilon(1-\eta)cv\Delta t. \tag{3}$$

When Eq. (3) is satisfied, we can ignore the term $d/v$ and further simplify $x$ as

$$x \le \min\left\{\frac{\eta}{1-\eta}, 1\right\}\frac{l}{k\Delta t}. \tag{4}$$

Then if $\eta < 1/2$, we have $x \le \frac{\eta l}{(1-\eta)k\Delta t}$.

#### 4.1.2 Partial-Charge Approach

In this approach, MC only needs to charge $\lambda$ sensors which are a portion of all sensors in each cluster. We have $1 \le \lambda \le (1-\eta)c$, since $\lambda$ cannot exceed the number of nodes with zero energy. Similar to the analysis above, we have

$$\begin{aligned} x\left(\lambda\Delta t + \frac{d}{v}\right) &\le \min\left\{\frac{\eta cl}{k}, \frac{\lambda l}{k}\right\}, \\ x &\le \min\left\{\frac{\eta c}{\lambda}, 1\right\}\frac{l}{k\Delta t}. \end{aligned} \tag{5}$$

If $\lambda/c \le \eta$, then we have $x \le \frac{l}{k\Delta t}$; otherwise, $x \le \frac{\eta cl}{\lambda k\Delta t}$.

When $\lambda/c \le \eta$, the upper bound of $x$ is $\frac{l}{k\Delta t}$, which is $\frac{1-\eta}{\eta}$ times greater than $x$ in Eq. (4). For example, if $\eta = 20\%$, MC

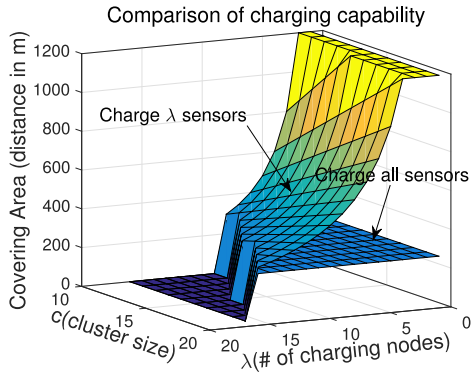



Fig. 3. Comparison of charging capability of MC by all-charge and partial-charge. $d = 60$ m, $l = 20$ h, $\Delta t = 0.5$ h, and $\eta = 20\%$, $k = 2$.

can cover a distance 4 times longer in a one-dimensional network using our approach. Therefore, we can utilize $\eta < 1/2$ and only charging a portion of zero energy sensors to extend the charging capability of MC while still satisfying the $k$-coverage requirement.

As an example, we compare the distance covered by MC for the two approaches in Fig. 3. We observe that the covering distance for charging $\lambda$ nodes scales much better than charging all nodes. For fixed cluster size $c$, the two approaches converge as $\lambda$ increases whereas more benefits are brought by charging only a small $\lambda$ number of nodes, e.g., the improvements can be as high as 4 times.

### 4.2 Covering Capability of MC in a Two-Dimensional Network

We now explore the condition under which the above result can be applied to a two-dimensional network (shown as the third case in Fig. 2).

For convenience, we plot clusters as circles with radius $r_s$ in Fig. 2. Recall that in Section 3, for $c$ sensors in a cluster of radius $r_s$, the upper bound of the shortest path is $2r_s(\sqrt{2c}+2)$. If it is less than $d$, then the traveling time within a cluster can be ignored. The result for a one-dimensional network can be used in a two-dimensional network when $2r_s(\sqrt{2c}+2) \leq d$, which is

$$r_s \leq \frac{d}{2(\sqrt{2c}+2)}. \tag{6}$$

By applying Eqs. (3) and (6), we obtain the maximum area a cluster can occupy, $2r_s \cdot d \leq [\epsilon(1-\eta)cv\Delta t]^2/(\sqrt{2c}+2)$. Since clusters and targets have a one-to-one mapping, its reciprocal yields a lower bound of target density $\rho_m$,

$$\rho_m \geq \frac{\sqrt{2c}+2}{[\epsilon(1-\eta)cv\Delta t]^2}. \tag{7}$$

If target density satisfies Eq. (7), in a two-dimensional network, the charging capability of MC in terms of maximum number of clusters it can handle can be calculated by applying the result in Section 4.1.

## 5 Forming Clusters for Targets

In this section, we consider how to form clusters to monitor the targets and also reduce the moving cost of MC. In general, in a $k$-coverage network, sensors in a larger cluster can share their workloads better and work less, whereas sensors in a smaller cluster work more and consume energy faster thereby requesting for charging more frequently. Our objective is to balance the number of sensors among neighboring clusters so that sensors would have similar loads and energy consumptions. In this way, a single charging round can cover more energy charging requests and reduce the moving cost of MC. We briefly describe how to form original clusters and select cluster heads. Then we investigate how sensors that can monitor multiple targets are assigned into clusters to balance cluster sizes.

### 5.1 Forming Cluster Considering Load Balance

A sensor can detect any target that is within its sensing range $r_s$. For those sensors that can only detect one target, we assign them to form the original clusters $\{C_i\}$. $C_i$ consists of the sensors that can only detect target $T_i$. In case that all the nodes around a target can also detect other targets, we initialize the cluster with a randomly picked node. A cluster head is selected for each cluster.

A number of algorithms have been proposed for cluster head selection in WSNs. In [9], a centralized algorithm is proposed to realize real-time head selection based on node concentration, energy level and centrality. In [10], a distributed algorithm is proposed to elect cluster head based on residual energy of nodes and the average energy of the network. We adopt the algorithm in [10] for cluster head election in our network. However, the randomness of target locations may result in overlapped clusters when targets are close. That is, a sensor can detect multiple targets and henceforth, may be assigned to multiple clusters. Next, we discuss how to resolve such contention and ensure that any node in the overlapped regions joins only one cluster and the variance of cluster size is reduced as much as possible.

The set of cluster heads is denoted as $\mathcal{H}$. Each head $H_i \in \mathcal{H}$ manages cluster $C_i$ which monitors target $T_i$. For brevity, we use the same subscript notation of clusters and their associated targets here interchangeably because they refer to the same cluster at the high-level. As an example, an (unclustered) sensor detecting multiple targets such as $T_a$ in $C_a$ and $T_b$ in $C_b$ $(a, b \in \mathcal{T}, \mathcal{C})$ is denoted by a tuple $[T_a, T_b]$. Tuples are distinguished according to their elements, we denote the $l$th type tuple as $\omega_l$. $|\omega_l|$ is the tuple size equal to the number of targets contained in $\omega_l$. The set $\{\omega_l\}$ is sorted in an ascending order regarding $|\omega_l|$, i.e., $|\omega_l| \leq |\omega_{l+1}|, \forall l$. To find nodes that can monitor the same targets, we group nodes having the same tuple $\omega_l$ into a set $\Psi_l$. All these different $\Psi_l$ form a bigger set $\{\Psi_l\}$. Next, we first give an example for handling tuples with $|\omega_l| = 2$.

At the beginning, a head $H_i$ initializes a *node count* $n_i$ as the number of sensors in the original cluster $C_i$. Then it progresses to examine tuples with $|\omega_l| = 2$. For head $H_i$, it needs to balance its cluster size by negotiating with the neighboring cluster head $H_j$. Both $H_i, H_j \in \mathcal{H}$ and the two clusters $C_i$, $C_j$ share $n_{ij}$ sensors. In other words, all these $n_{ij}$ nodes can detect and only detect targets $i$ and $j$. Assume that the current node count in cluster $C_j$ is $n_j$. Without loss of generality, assume $n_i \leq n_j$, the case for $n_i > n_j$ can be similarly handled. If $n_i + n_{ij} \leq n_j$, then assign all shared $n_{ij}$ nodes to $C_i$; if $n_i + n_{ij} > n_j$, then assign $\lfloor (n_i + n_j + n_{ij}/2) \rfloor - n_i$ sensors to $C_i$, and the rest of shared sensors to $C_j$.

For tuples with $|\omega_l| = 2$, the algorithm proceeds to examine all pairs of neighboring cluster heads $H_i, H_j \in \mathcal{H}$. Next, a new round for tuples with $|\omega_l| = 3$ is initiated

TABLE 2
Cluster Size Balancing Algorithm

**Input**: Set of different types of tuples $\{\omega_l\}$;
sets of nodes having the same tuple $\omega_l$ denoted as $\{\Psi_l\}$;
original clusters $\{C_i\}$ consisting of sensors only detecting $T_i$;
number of sensors in original cluster $C_i$ denoted as $n_i$.
**Output**: Set of size-balanced clusters $\{C_i\}$.
**for** $l = 1, 2, \ldots, |\{\Psi_l\}|$
**while** $\Psi_l \neq \phi$
Node $u \in \Psi_l$;
$k \quad \arg\min_{i \in \omega_l}\{n_i\}$;
$C_k \quad C_k \bigcup \{u\}$; $n_k \quad n_k + 1$;
$\Psi_l = \Psi_l \setminus \{u\}$.
**end while**
**end for**

and the iteration goes on until all the nodes in the overlapped regions have been assigned to appropriate clusters. The algorithm keeps the sensors with larger tuple size for later assignments since they have more flexibility to join clusters compared with the sensors that only detect fewer targets.

In general, the cluster size balancing algorithm is described as follows. We go through the set $\{\Psi_l\}$ from $\Psi_1$. Among the targets in tuple $\omega_1$, we find target $T_k$ corresponding to the cluster $C_k$ with minimum $n_k$. A node $u$ is randomly picked from $\Psi_1$, and assigned to cluster $C_k$. Thus, $n_k$ is increased by 1. Then we remove node $u$ from $\Psi_1$. For the remaining nodes in $\Psi_1$, we repeat the above process until $\Psi_1$ is empty. The iteration continues for $\Psi_2$, $\Psi_3$, ..., until the set $\{\Psi_l\}$ is exhausted. This algorithm is summarized in Table 2.

Fig. 4 illustrates the cluster size balancing algorithm by an analogy of placing balls (sensors) into bins (clusters). Here, we have 3 clusters, $C_1, C_2, C_3$ with 4 overlapped regions. Nodes in the overlapped regions are colored in different colors. The algorithm starts from the overlap between $C_1$ and $C_2$, which shares 3 red balls. Note that red balls are only shared by $C_1$ and $C_2$, thus they cannot be assigned to $C_3$. After balancing, $C_1$ and $C_2$ are assigned 2 and 1 red balls respectively, so the updated numbers of balls (sensors) in $C_1$ and $C_2$ become 5 and 6. Similarly, the next steps distribute green balls and yellow balls shared by $C_1$, $C_3$ and $C_2$, $C_3$, respectively. Finally, the purple ball shared by all three bins together is assigned to $C_1$ since $C_1$ has the least number of balls. We can see that after the balancing, the numbers of nodes in $C_1, C_2, C_3$ become 8, 7, and 8, respectively, which are close to each other.

We now analyze the message overhead of the above algorithm. Note that for $m$ clusters, although there may be $\binom{m}{2} + \binom{m}{3} + \cdots + \binom{m}{m} = 2^m - m$ enumerations of overlapped regions, the actual number of iterations is bounded by the number of nodes $n$ in the overlapped region. In each iteration, only a constant number of messages are exchanged, and the number of iterations is at most $n$, so the overall message overhead is $\mathcal{O}(n)$.

### 5.2 Spatial Characteristics of Communication Links

The actual communication range is related to the dynamics of wireless channel and physical environment. In practice, there could be cases that a further node has better communication link. According to [17], such factors are empirical and difficult to predict. Hence, for analytical tractability, we follow a general and classic approach in [18], which models the receptive energy in proportions to $d^{\alpha}$. $\alpha$ is a path loss exponent ranges from $-2$ to $-4$. In implementations, we could substitute this theoretical model to a practical one based on measured signal levels, whereas the problem formulation and algorithms proposed in this paper should still work.

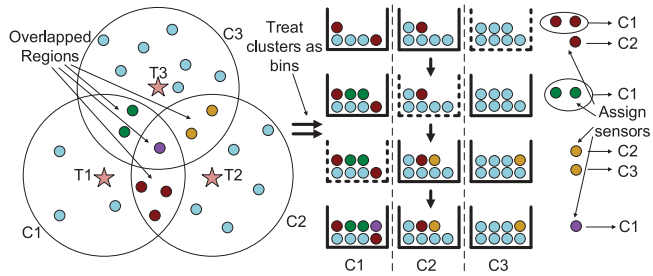



Fig. 4. An example of balancing nodes in neighboring clusters.

## 6 Charging Scheduling of Mobile Charger

In this section, we study the charging scheduling of MC in a target $k$-coverage network. First, we propose a new distance metric by jointly considering traveling distance and cluster lifetime. Second, we find the shortest Hamilton path through clusters by transforming GTSP to TSP. Based on the charging sequence, we formulate the problem into an *Integer Programming* problem to maximize the number of charged nodes $(\lambda_i, i \in \mathcal{C})$ in each cluster per unit lifetime. Finally, we derive the so called $\lambda$-GTSP charging route, and give an example to demonstrate the complete process in this section. The above steps are altogether summarized as the $\lambda$-GTSP Charging Algorithm.

### 6.1 New Distance Metric

Calculating the charging sequence without taking node lifetime into consideration may easily lead to infeasible solutions. Intuitively, to maximize performance, MC needs to charge as many nodes as possible. However, charging more sensors at the beginning of the sequence would inevitably elongate the entire charging process and postpone charging for subsequent nodes. These nodes may deplete energy before MC arrives, which violates the target $k$-coverage requirement. On the other hand, to meet battery deadlines from all clusters, MC may visit only one node from each cluster $(\lambda_i = 1, \forall i \in \mathcal{C})$. Nevertheless, this scheme is inefficient due to high moving cost, since MC has to come back again for other charging requests eventually. Therefore, our goal is to find a balance in between.

Consider a set of clusters $C_1, C_2, \ldots, C_q$ sending charging requests. A cluster with limited lifetime later in this sequence is a bottleneck since all the clusters ahead need to reduce their charged nodes number $\lambda_i$ until the charging time spent on them no longer violates the charging schedule for the bottleneck cluster. A natural solution is to push the bottleneck forward in the charging sequence so its impact on the remaining clusters is minimum. Hence, we introduce a new distance metric $d'_{uv}$

$$d'_{uv} = d_{uv}\frac{l_i l_j}{L_i L_j} = d_{uv}\frac{l_i l_j}{\lfloor \frac{n_i}{k} \rfloor \lfloor \frac{n_j}{k} \rfloor l^2}, \tag{8}$$

where $d_{uv}$ is the Euclidean distance between nodes $u$ and $v$, which belong to clusters $C_i$ and $C_j$ separately. $l_i/L_i$ and $l_j/L_j$ are the normalized lifetime for clusters $C_i$, $C_j$, where $l_i$ is the remaining lifetime and $L_i$ is the maximum lifetime of cluster $C_i$. For example, if $l_i = L_i$, $l_j = L_j$, the new distance

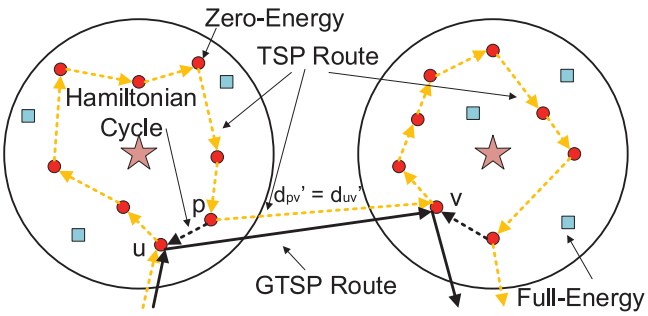



Fig. 5. Derivation of GTSP route by solving TSP.

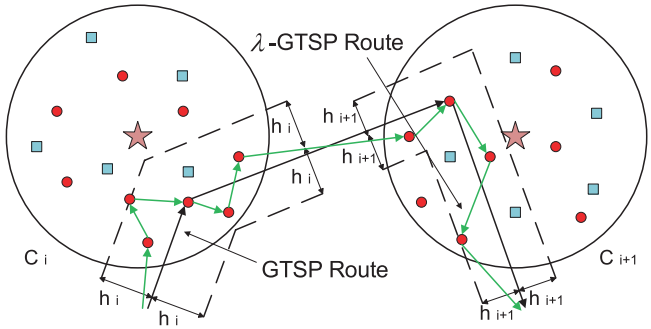



Fig. 6. Derivation of $\lambda$-GTSP charging route.

$d'_{uv} = d_{uv}$, whereas if $l_i = L_i/2, l_j = L_j/2$, $d'_{uv} = d_{uv}/4$. If two nodes in the field have shorter lifetime, their "distance" is also much smaller. Thus, during tour planning, the edges between nodes with less $d'_{uv}$ would be considered with higher priority, and clusters of shorter lifetime can be visited earlier by MC. Next, we develop such a tour planning algorithm based on this new distance metric.

### 6.2 Transforming GTSP to TSP

Recall the definition in Section 3, solving *Generalized Traveling Salesmen Problem* (GTSP) for the target $k$-coverage network gives the shortest route which visits exactly one node in each cluster. In the following two Sections 6.3 and 6.4, charging route of MC is derived based on the GTSP route calculated in this section.

To find the shortest route through clusters, we transform GTSP into a TSP so that we can apply classic TSP algorithms (e.g., nearest neighbor) for the problem. The transformation process is based on the algorithm in [19]. Fig. 5 demonstrates an example of the algorithm. First, a set of arbitrary Hamiltonian cycles are formed in each cluster. The Hamiltonian cycle starts from any selected sensor, "visits" all the nodes with zero energy exactly once and returns to the starting sensor. The direction is picked arbitrarily (clockwise or counterclockwise), and henceforth, each node has its direct parent node. Second, the distances along the Hamiltonian cycle is set to zero. As shown in Fig. 5, $d'_{pu} = 0$ since $p$, $u$ are adjacent in the same Hamiltonian cycle. For nodes $u$ and $v$ in different clusters, we set the distance from $u$'s direct parent node $p$ to $v$, $d'_{pv} = d'_{uv}$. After the above steps, solving GTSP has been transformed into solving TSP. Finally, based on these distances, we run a TSP algorithm on all clusters to form the shortest path that traverses all zero-energy nodes in each cluster, as denoted as the yellow dashed lines in the figure. The route has one entering node $u$ and one exiting node $p$ in each cluster. The exiting node $p$ is the parent node of the entering node $u$. Since we have set $d'_{pv} = d'_{uv}$, $u$ is selected as the only charging node in its cluster, which solves the GTSP problem.

### 6.3 Optimizing $\lambda_i$

We further find the optimal number of nodes to be charged in each cluster $C_i$. By solving GTSP, we obtain a charging sequence of clusters $\{C_1, C_2, \ldots, C_q\}$. For convenience, denote the traveling distance between two consecutive clusters $C_{i-1}$, $C_i$ as $d_i$. The lifetime $l_i$ of cluster $C_i$ can be found based on $k$ and residual energy of sensors,

$$l_i = \left\lfloor \frac{|F_i|}{k} \right\rfloor \cdot l + \frac{\sum_{j \in W_i} E_j}{k c_s}, \tag{9}$$

where $F_i$ is the set of sensors in sleeping mode with full energy, $W_i$ is the set of sensors in working mode, $c_s$ is average energy consumption rate of a working sensor, $E_j$ is residual energy of node $j$. We formulate the optimization problem into an Integer Programming with the objective of maximizing the total number of nodes charged per unit lifetime over all clusters

$$\mathbf{P1}: \quad \max \sum_{i=1}^{q} \frac{\lambda_i}{l_i} \tag{10}$$

Subject to

$$\sum_{i=1}^{j-1} \lambda_i \Delta t + \sum_{i=1}^{j} d_i/v \le l_j, \forall j = 2, 3, \ldots, q, \tag{11}$$

$$1 \le \lambda_i \le N_i^0, \forall i = 1, 2, \ldots, q, \tag{12}$$

where $N_i^0$ is the number of nodes that have depleted energy in cluster $C_i$. $\lambda_i$ in Eq. (10) can be considered as the benefits brought by charging, which is further scaled by the reciprocal lifetime of a cluster $1/l_i$. In this way, if the lifetime of a cluster is low, charging more sensors in it would bring more benefits and charging clusters with long lifetime would bring fewer benefits. We design the objective function in this way so that limited resources from MC can be distributed better among different clusters. In addition, Eq. (11) states that all prior charging time plus traveling time to a cluster should be less than its lifetime so it can guarantee $k$-coverage. Eq. (12) stipulates that MC charges at least one and a maximum of $N_i^0$ nodes.

Since Integer Programming is NP-complete [27], a simple way is to adopt Linear Programming relaxation and round the results to a smaller integer (take the floor operation at the end). In our case, the efficient Integer Programming solver CPLEX is used for deriving $\lambda_i$ [28].

### 6.4 Calculating $\lambda$-GTSP

After $\lambda_i$ has been calculated for each cluster, MC selects which $\lambda_i$ nodes should be added into the charging routes such that the additive moving distance to the original GTSP route is minimal. To prevent MC from deviating from the GTSP route, a sweeping sector is created by two lines that are parallel to the moving trajectory of MC (as shown in Fig. 6). The sweeping sector is gradually expanded to add more nodes with zero energy until the sector contains $\lambda_i$ nodes and TSP is solved in each cluster to connect $\lambda_i$ picked nodes by a shortest path. Finally, a $\lambda$-GTSP charging route is generated which traverses through $\lambda_i$ nodes in each cluster calculated by the IP in Eq. (10).

TABLE 3
$\lambda$-GTSP Charging Algorithm

**Input**: A number of $q$ simultaneous energy requests; distance $d'_{uv}$ between nodes $u, v$; set of all the clusters $\{C_i\}$; lifetime of cluster $C_i$ denoted as $l_i$.
**Output**: $\lambda$-GTSP charging route for MC.
**Step 1**: Construct (directed) Hamiltonian cycle in each cluster; for adjacent nodes $p$, $u$ in the same Hamiltonian cycle, $d'_{pu}$ 0;
for nodes $p$, $v$ from different clusters, $p$ is parent of $u$ , $d'_{pv} \leftarrow d'_{uv}$.
**Step 2**: Based on new $d'_{uv}$, solve *TSP*, obtain *GTSP* route $\bigcup_i$ {entering point of TSP route in $C_i$};
charging sequence $\leftarrow \{C_1, C_2, \ldots, C_q\}$.
**Step 3**: Solve Integer Program **P1**: Eqs. (10), (11), (12), obtain $\lambda_i$.
**Step 4**: Width of sweeping sector $h_i = 0$; $\delta$ = width increment;
for all the clusters, **do**
  $h_i$ $h_i + \delta$, construct sweeping sector of width $h_i$;
  $Z_i$ set of zero-energy sensors in sweeping sector;
**until** $|Z_i| = \lambda_i$, $\forall i = 1, 2, \ldots, q$.
Obtain $\lambda$-GTSP route by solving TSP over $\bigcup_i \{Z_i\}$.

An example of the $\lambda$-GTSP charging algorithm is shown in Fig. 6. A segment of the GTSP charging route is depicted by the black arrows from cluster $C_i$ to $C_{i+1}$. We focus on the red nodes that have depleted energy. Assume that solving the Integer Programming **P1**: Eqs. (10), (11), (12) gives a solution $\lambda_i = 5$ and $\lambda_{i+1} = 4$. The sweeping sectors are contained within the dashed lines ($h_i$ and $h_{i+1}$ distances away from the GTSP trajectory). Note that $h_i$ and $h_{i+1}$ could be different since the processes are performed independently in different clusters. We denote the set of zero-energy sensors within the sweeping sector for $C_i$ as $Z_i$. $h_i$ increases from 0 by $\delta$ each time until $\lambda_i$ is reached ($|Z_i| = 5$ and $|Z_{i+1}| = 4$ for $\lambda_i = 5$, $\lambda_{i+1} = 4$). The shortest Hamiltonian path through all these $\lambda_i$ nodes in $C_i$ is the $\lambda$-GTSP route represented by the green arrows in Fig. 6.

The above calculations can be done at the base station and disseminated to MC through long range wireless communications such as LTE. We summarize all the above process discussed in Section 6 as $\lambda$-GTSP Charging Algorithm in Table 3.

### 6.5 An Example of $\lambda$-GTSP Network

We demonstrate the process of $\lambda$-GTSP algorithm in Fig. 7 by taking a snapshot during the operation. We set the number of targets to 7 in a square field comprised of 60 sensors. The sensors within sensing range $r_s$ of targets are organized into clusters by the distributed cluster size balancing algorithm in Section 5. Upon receiving charging requests from the clusters, base station derives the GTSP route denoted as the blue lines in Fig. 7a based on the new distance metric. Finally, the $\lambda$-GTSP algorithm calculates number of charged nodes $\lambda_i$ for each cluster and iteratively finds those sensors along the original GTSP trajectory and the corresponding $\lambda$-GTSP route in Fig. 7b.

### 6.6 Optimizing the Selection of Working Sensors

In the previous discussions, $k$ alive sensors are randomly picked and turned into working mode in each cluster. However, this method may incur extra traveling cost on the MC since the selection is random. Based on this observation, an algorithm named $K$-means Working Sensors Determination Algorithm is proposed by applying $K$-means clustering [29]

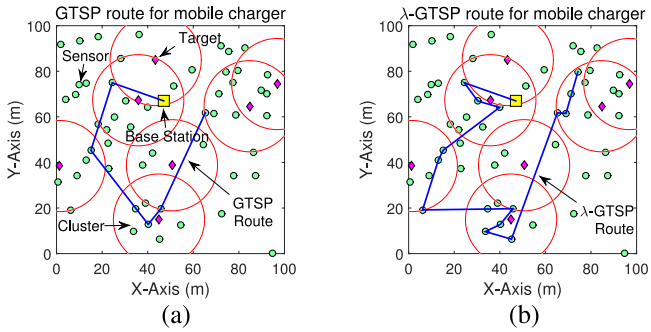



Fig. 7. A running example of $\lambda$-GTSP network. (a) GTSP route for mobile charger. (b) $\lambda$-GTSP route for mobile charger.

to determine which sensors should be woken up. The algorithm aggregates sensors in proximity into some sub-clusters, and sensors within one sub-cluster are turned on at first, which reduces the movement of MC during charging operations. To avoid ambiguity between $K$-means and $k$-coverage, capital $K$ is used here in $K$-means. By applying $K$-means algorithm in each cluster, the sensors are divided into a set of $K$ sub-clusters $\mathcal{B} = \{B_1, B_2, \ldots, B_K\}$, so that the intra-cluster sum of squares (i.e., variance) is minimized,

$$\arg\min_{\mathcal{B}} \sum_{i=1}^{K} \sum_{x,y \in B_i} ||x - y||^2. \quad (13)$$

$x$ and $y$ are the positions of any two sensors in the sub-cluster $B_i$. Here, partitioning is based on the euclidean distance and each cluster is further divided into several sub-clusters. Utilizing this property, $k$ working sensors can be picked from one sub-cluster until all the sensors deplete their energy in the sub-cluster. Within a sub-cluster, the shortest path through all the sensors is found by solving a TSP. Then $k$ sensors are woken successively along the path. If the energy of all sensors within a sub-cluster has depleted, the nearest sensor in another sub-cluster with full battery will be picked. A TSP is solved again in the new sub-cluster, and another shortest path is formed, where sensors are chosen in the same way along the path. We can see that rather than random selections, the sensors with energy depletion can aggregate so that additional moving cost from the MC can be saved. Partitioning is conducted one-time at the beginning, and the sub-clusters remain fixed in the rest of the process. After charging is completed in one cluster, $k$ sensors need to be picked to start a new round of target monitoring. To aggregate energy-depleted sensors, the sub-cluster that has the largest number of energy-depleted sensors is chosen and the process repeats. The algorithm is summarized in Table 4.

For the complexity of the algorithm, a popular approximation algorithm called Lloyd's Algorithm is applied one-time, whose complexity is $O(nKi)$ [30], where $i$ is the number of iterations. For each round of charging, TSP is solved for each sub-cluster. Since TSP is NP-hard, the well-known Nearest Neighbor (NN) algorithm is applied as a heuristic algorithm with good empirical performance, whose time complexity is $O(n^2)$ [31]. Applying NN algorithm $K$ times for the $K$ sub-clusters in cluster $C_i$ takes $O(n_i^2)$. For the whole field, the time complexity of solving TSP is $O(n^2)$.

### 6.7 Interaction of Route and $\lambda_i$

The choice of $\lambda_i$ has an impact on the charging route, and vice versa. A two tier scheduling scheme concentrates

TABLE 4
$K$-Means Working Sensors Determination Algorithm

**Input**: Cluster $C$; Parameter $K$.
**Output**: Sequence of $k$ working sensors in $C$.
Solve $K$-means clustering in $C$,
Get set of sub-clusters $\mathcal{B} \leftarrow \{B_1, B_2, \ldots, B_K\}$.
$s \quad \arg\max_{1 \le j \le |\mathcal{B}|} |\{x : x \text{ is depleted and } x \in B_j\}|$.
**Start:** Run TSP for $B_s$, get the shortest path $\mathcal{P}_s$.
**While** $|\mathcal{P}_s| \ge K$
Activate the first $k$ sensors in $\mathcal{P}_s$.
**Until** $k$ sensors deplete energy,
Remove them from $\mathcal{P}_s$ and $B_s$.
**End Until**
**End While**
$\mathcal{P} \leftarrow \mathcal{P}_s$, $u \leftarrow$ the last node in $\mathcal{P}_s$;
$s \leftarrow \arg\min_{1 \le j \le |\mathcal{B}|, j \ne s, x \in B_j} ||x - u||$.
Activate $k - |\mathcal{P}|$ sensors in $B_s$; Remove them from $B_s$.
**Return** to **Start**

energy consumption to those sensors to be reclaimed by a Mobile Repairman (MR) [37]. However, this method may not be readily applicable in our circumstance since energy consumption of sensors cannot be fully controlled, i.e., it is determined by the real-time position of targets and the value of $k$-coverage. The dispatch of mobile chargers relies on the energy consumptions of sensors. Reducing the interaction between energy consumption and route planning could extend the network lifetime. We take the following several steps to achieve this. First, sensors with charging requests are restricted within the sensing range of each target. Therefore, the extra distances traveled due to the choices of charged sensors is restricted to a low level. Second, k working sensors are chosen according to K-means clustering within each cluster so that the charging requests are concentrated. Third, the $\lambda$-GTSP route is modified from the original GTSP route, which is the shortest route connecting the clusters sending charging requests. Thus, the route derived from $\lambda$-GTSP is close to the optimal solution. In the future, we plan to find the optimal charging strategy considering the interactions between them.

## 7 Mobile Targets

In this section, we extend our algorithms to handle mobile targets and demonstrate that it only requires minimum changes. On the other hand, since the mobility pattern of targets is dynamic, for effectiveness, we allow the network to re-cluster at certain points. Thus, we give the condition on when such re-clustering is needed.

### 7.1 Cluster Expansion

In practice, there are a growing number of applications that require sensors to not only monitor stationary targets but also mobile targets. For example, imaging sensors can utilize machine learning algorithms to detect pedestrian, and environmental sensors deployed in the habitat can monitor migration patterns of animals. Since the mobility patterns are specific in different applications, we present a general study when targets have random mobility and model their movements by 2-dimensional *Brownian motion*. After the first discovery of random motion of particles in fluid, Brownian motion finds many applications in the field of physics, finance and engineering [15]. In [16], the

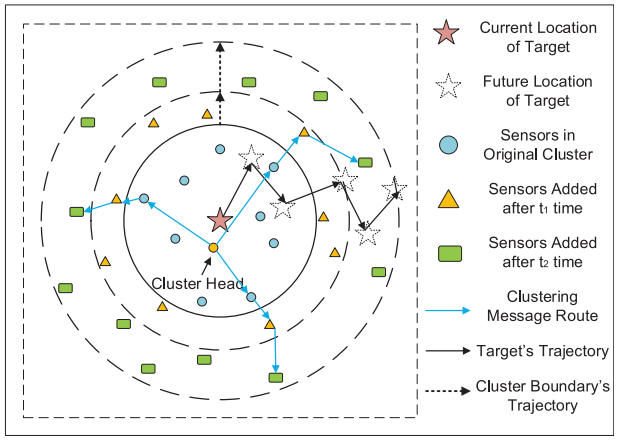



Fig. 8. Dynamic reclustering by extension of cluster radius.

moving trajectories of targets in a WSN are characterized by Brownian motion. In such a model, a target has equal chances to move towards any direction. The model is memoryless such that the preceding step of a target is independent of its succeeding step.

Since sensors are stationary, we need to know the deviation of target positions regarding their initial positions over time. It is represented by the *mean square displacement*,

$$< x^2 > = 2 \cdot dim \cdot Dt, \tag{14}$$

where *dim* is the dimensions of the space (2 for the 2-dimensional sensing field), and $D$ ($\mathrm{m}^2/\mathrm{s}$) represents the diffusion coefficient of targets, and $t$ is a time span. The diffusion coefficient $D$ is proportional to the moving speed of a target so a higher $D$ corresponds to faster speed. In this paper, we have assumed the targets are identical so they have the same diffusion coefficient $D$. The square root of mean square displacement represents the expected displacement of target over time. For a 2-dimensional field, $\sqrt{< x^2 >} = 2\sqrt{Dt}$.

Based on the expected displacement of targets, clusters should expand their boundaries accordingly to cover targets with high probability. For static targets, the boundary of a cluster is fixed (at a maximum of $r_s$). For mobile targets, after observing a target for $t$ time, the expected displacement is $2\sqrt{Dt}$ so this value should be added to $r_s$ as the new radius of cluster. It can be done by the cluster head to propagate a message at time $t$ to the nodes that are $h = \lceil (r_s + 2\sqrt{Dt})/r \rceil$ hops away, where $r$ is the transmission range of sensors. That is, a node outside the original cluster falls into $h$-hop communication range after some time $t$ and this node has not joined any cluster yet. It will join at time $t$ to monitor the target if the expanded boundary reaches the node. An example of cluster expansion is demonstrated in Fig. 8.

### 7.2 Re-Clustering Condition

As we have seen, to maintain $k$-coverage of a mobile target, the cluster should expand to add more sensors for monitoring. Such expansion cannot continue forever due to following reasons. First, the cluster boundary at time $t$ only represents the expected target displacement, it is possible that the target has already moved out of the cluster. Second, the number of sensors that participate in monitoring the moving target also increases to assure target $k$-coverage. For the original cluster of area $\pi r_s^2$, the number of working

sensors per unit area is $k/(\pi r_s^2)$. For uniform sensor distributions, this number is increased to $(r_s + 2\sqrt{Dt})^2 k/r_s^2$ at $t$ to maintain the density of working sensors in expanded cluster. The expansion should stop before MC can no longer cover all charging requests in the sensing field. Re-clustering should be initiated when either one of the two conditions is met. Next, we derive the condition for re-clustering.

As discussed in Section 4, if the target density $\rho_m$ in an area satisfies Eq. (7), then the maximum number of targets that one MC can cover is $l/(k\Delta t)$. For the expanded cluster here, $k$ should be replaced by $(r_s + 2\sqrt{Dt})^2 k/r_s^2$. Thus, the quantity of targets that one MC can handle should be greater than or equal to the total number of targets in the field $\mathcal{A}$.

$$\frac{l r_s^2}{(r_s + 2\sqrt{Dt})^2 k \Delta t} \geq m. \tag{15}$$

Solving the equation, we derive the upper bound of $t$ for re-clustering. Combining with the first case, the re-clustering time is denoted as

$$t_{re} = \min\left\{\tau, \frac{r_s^2}{4D}\left(\sqrt{\frac{l}{km\Delta t}} - 1\right)^2\right\}, \tag{16}$$

where $\tau$ is the time when the target moves out of cluster; otherwise, $\tau = \infty$. Compared with re-clustering time, re-clustering frequency $f_{re}$ is more intuitive, which is the reciprocal of the re-clustering time and denoted as,

$$f_{re} = \max\left\{\frac{1}{\tau}, \frac{4D}{r_s^2}\frac{1}{(\sqrt{\frac{l}{km\Delta t}} - 1)^2}\right\}. \tag{17}$$

$f_{re}$ represents the times of re-clustering in unit time. For example, if the re-clustering time is 120 days, then the re-clustering frequency is 0.25 times/month.

### 7.3 Brownian Motion with Drift

In the above sections, we assumed that the motion pattern of targets follows pure Brownian motion. In practice, targets may exhibit other patterns, such as heading towards a specific direction. For example, animals could be attracted by food sources, and move randomly with a certain drift towards the source [32]. Forest fire may be driven by the speed and direction of the wind [33]. These patterns can be modeled by Brownian motion with drift [34]. Here, the animals or phenomena are the targets to monitor in our model. The displacement of such target can be represented as follows,

$$X_t = \mu t + \sigma Z_t, \tag{18}$$

where $t$ is the time, $\mu$ is the drift velocity, $\sigma$ is the scale parameter, and $Z_t$ is a standard Brownian motion random variable [34]. The variance of Brownian motion with drift and the variance of the standard Brownian motion are equivalent. Since $< X_t^2 > = Var(X_t) + < X_t >^2$, and Eq. (14) shows that $< X_t^2 > = 2 \cdot dim \cdot Dt$, the *mean square displacement* can be represented by,

$$< X_t^2 > = 2 \cdot dim \cdot Dt + (\mu t)^2. \tag{19}$$

For a 2-dimensional field, the expected displacement is the square root of the mean square displacement ($\sqrt{4Dt + (\mu t)^2}$). By the same token, the radius of the cluster associated with the target should extend to $r_s + \sqrt{4Dt + (\mu t)^2}$ at time $t$. For clarity, let $y = (\mu t)^2$, the new re-clustering time is,

$$t_{re} = \min\left\{\tau, \frac{r_s^2}{4D}\left(\sqrt{\frac{l}{km\Delta t}} - 1\right)^2 - \frac{y}{4D}\right\}, \tag{20}$$

where $\tau$ is the time when the target moves out of cluster; otherwise, $\tau = \infty$. The associated re-clustering frequency is,

$$f_{re} = \max\left\{\frac{1}{\tau}, 4D \cdot \frac{1}{r_s^2(\sqrt{\frac{l}{km\Delta t}} - 1)^2 - y}\right\}. \tag{21}$$

Comparing Eq. (20) with the re-clustering time for Brownian motion as shown in Eq. (16), the second term in the bracket decreases by $y/(4D)$. Since $y$ is proportional to the square of time and drift velocity, it declines rapidly. It leads to a large drop of the re-clustering time and a large increase of the re-clustering frequency and makes the re-clustering message overhead too large to handle. This observation necessitates a new re-clustering strategy to offset the decrease of re-clustering time.

TABLE 5
Dynamic Clustering Algorithm for Brownian Motion with Drift

**Input**: Initial radius $r_s$ of cluster; time increment $\delta$
Set of sensors $\mathcal{N}$, set of targets $\mathcal{T}$, set of position of sensors $\vec{P}$; diffusion coefficient $D$ of target, drift velocity $\mu$ of target.
**Output**: Dynamic clusters for each target changing with time $t$.

Re-clustering time $t_{re}$ $\quad \frac{r_s^2}{4D}\left(\sqrt{\frac{l}{km\Delta t}} - 1\right)^2$;
**While** $t < t_{re}$,
  $t \leftarrow t + \delta$;
  **For** $\forall i \in \mathcal{T}$, **If** $i$ moves out of $C_i$, **Then** $\mathcal{T} \setminus \{i\}$;
    $\forall j \in \mathcal{N}$, $d_{ij} \leftarrow |\vec{P}_j - (\vec{P}_i + \mu t)|$;
  **If** $d_{ij} \leq r_s + 2\sqrt{Dt}$
    **Then** $C_i \quad C_i \bigcup \{j\}$
  **If** $d_{ij} > r_s + 2\sqrt{Dt}$, **and** $j \in C_i$;
    **Then** $C_i \quad C_i \setminus \{j\}$.
  **End For**
**End While**

### 7.4 Probabilistic Dynamic Clustering Algorithm

In this subsection, we propose a new dynamic clustering algorithm, and study the message overhead and re-clustering time.

#### 7.4.1 Clusters with Drifting Motions

The new scheme extends the algorithm proposed in Section 7.1 and adapts to motion patterns with other probabilistic models. Previously, cluster boundary expands with a fixed center, which may not cover the target with a drift velocity. This observation motivates us to introduce mobility to the cluster as well. It allows the cluster to move at the same velocity $\mu$ along the same direction with targets. The radius of the new cluster is $r_s + 2\sqrt{Dt}$ and the centroid is $\mu t$ distance away from the original centroid at time $t$. In practice, a message with the original position, diffusion coefficient and drift velocity of the target is flooded in the network. The sensor receiving the message calculates its distance from the moving centroid, and determines whether it is included in the cluster or not. The new scheme is summarized in Table 5.

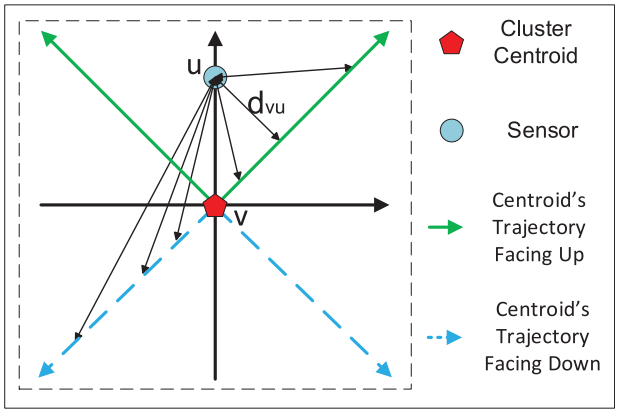



Fig. 9. Two cases of distance $d_{vu}$ between cluster centroid and sensor evolving with time.

Cluster mobility counteracts the impact of the drift velocity from targets. Thus, the new re-clustering time is $t_{re} = \min\left\{\tau, \frac{r_s^2}{4D}\left(\sqrt{\frac{l}{km\Delta t}} - 1\right)^2\right\}$, that is the same with the re-clustering time derived in Section 7.2. The new scheme can be extended to other motion patterns of diverse probabilistic models. From probability distributions, expected target movement at $t$ is $< X_t >$ and the centroid of the new cluster at $t$ is set to $< X_t >$ distance away from the original position. The variance of target movement at time $t$ is $< X_t^2 > - < X_t >^2$ and the radius of the new cluster at time $t$ is set to $r_s + \sqrt{< X_t^2 > - < X_t >^2}$. The re-clustering time can be derived similarly as shown in Section 7.2.

To make sure $k$ sensors are available around the target, we need to increase the number of working sensors in a cluster from $k$ to $(r_s + \sqrt{< X_t^2 > - < X_t >^2})^2 k / r_s^2$. Since the maximum number of targets covered by one MC is $l/(k\Delta t)$, the new upper bound of MC's covering capability can be derived by replacing $k$ with $(r_s + \sqrt{< X_t^2 > - < X_t >^2})^2 k / r_s^2$. It needs to be greater than the total number of targets,

$$\frac{l r_s^2}{(r_s + \sqrt{< X_t^2 > - < X_t >^2})^2 k \Delta t} \geq m. \quad (22)$$

Eq. (22) provides a lower bound of re-clustering time given the probability distribution.

### 7.4.2 Message Overhead of the Original Clustering

We analyze the message overhead of the original clustering strategy. At the beginning, the cluster head broadcasts a timestamped message to surrounding sensors, which includes the current location of the target, the radius $r_s$ of the cluster, and the expected diffusion coefficient $D$ of the target. When a sensor $u$ outside of the cluster receives this message, it calculates when it needs to join the cluster by $(d_{vu} - r_s)^2/(4D)$, where $d_{vu}$ is the distance between target $v$ and sensor $u$. After $u$ joins the cluster, it broadcasts a message to the surrounding sensors to inform about its position and cluster. Before re-clustering, the number of sensors in the cluster is bounded by $\rho\pi(r_s + 2\sqrt{Dt_{re}})^2$. For a field of area $S$, the density of sensors is $\rho = n/S$. For each sensor, only a constant number of messages are sent during cluster expansion. Therefore, the message overhead of the original clustering strategy is $\mathcal{O}(n(r_s + 2\sqrt{Dt_{re}})^2)$.

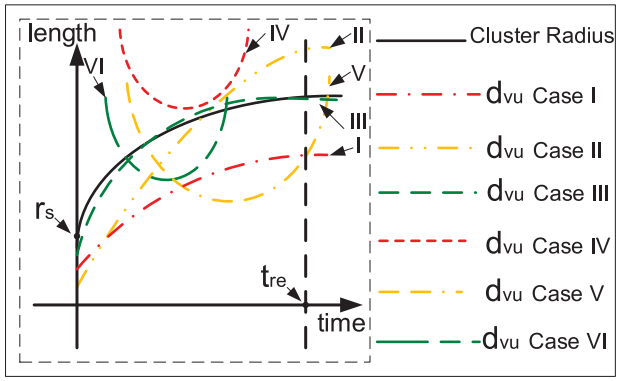



Fig. 10. Different cases of relative locations of $d_{vu}$ and cluster radius.

### 7.4.3 Clustering Message Overhead of the Probabilistic Dynamic Clustering

Next, we analyze the message overhead of the new clustering strategy. At the beginning, the cluster head broadcasts a message to surrounding sensors with an additional value of the drift velocity $\mu$. The sensor outside of the cluster joins the cluster at time $(d_{vu} - r_s)^2/(4D)$. Different from the previous clustering algorithm, $d_{vu}$ is now changing regarding $t$, where $d_{vu} = ||\vec{P}_u - (\vec{P}_v + \vec{\mu}t)||$. $\vec{P}_u$ is the position vector of the sensor $u$, $\vec{P}_v$ is the position vector of the target at the beginning and $\vec{\mu}$ is the drift velocity vector of the target. $d_{vu}$ is compared with the cluster radius $r_s + 2\sqrt{Dt}$ to determine whether $u$ should be included in the expanded cluster. If $d_{vu}$ is smaller than $r_s + 2\sqrt{Dt}$, then the sensor $u$ should be included; otherwise, it should be removed.

**Property 1.** *The total number of messages sent by any sensor in probabilistic dynamic clustering is bounded by a constant.*

**Proof.** As shown in Fig. 9, if the trajectory of the cluster centroid is located in the region above the horizontal axis, i.e., the solid green trajectories, then $d_{vu}$ decreases first and then increases. In this case, the curves of $d_{vu}$ are shown by the convex curves (i.e., Cases I-III) in Fig. 10. If the trajectory of the target is located in the region below the horizontal axis, i.e., the dashed blue trajectories, then $d_{vu}$ is monotonically increasing. In this case, the curves of $d_{vu}$ are shown by the concave curves (i.e., Cases IV-VI) in Fig. 10.

There are 6 cases of the sensor's affiliations with the cluster as shown in Fig. 10, based on the relative values of $d_{vu}$ and cluster radius. For Case I, one sensor included in the starting cluster remains included in the cluster before the re-clustering time $t_{re}$. For Case II, one sensor included in the starting cluster is removed from the cluster at some time and never rejoins the cluster before $t_{re}$. For Case III, one sensor included in the starting cluster is removed from the cluster and rejoins the cluster in later time before $t_{re}$. For Case IV, one sensor not included in the starting cluster never joins the cluster before $t_{re}$. For Case V, one sensor not included in the starting cluster joins the cluster at some time and never leaves the cluster before $t_{re}$. For Case VI, one sensor not included in the starting cluster joins the cluster at some time and is removed from the cluster in a later time before $t_{re}$.

The radius of the cluster is a monotonic increasing curve denoted by a solid black line. It is a concave curve

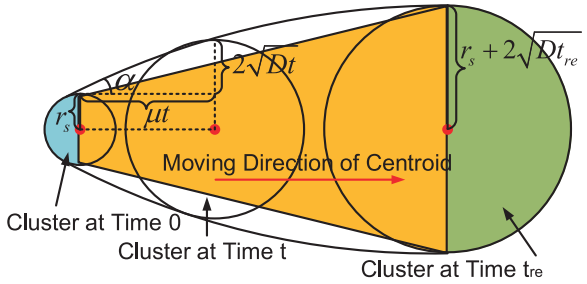



Fig. 11. Area covered by the clusters in the probabilistic dynamic clustering.

since the increasing rate of radius of the cluster is governed by $\sqrt{t}$. If the curve of $d_{vu}$ is below the curve of cluster radius, then the corresponding sensor should be included in the cluster; otherwise, it should be excluded from the cluster. In Fig. 10, Cases I-III denoted by concave curves correspond to the trajectories of the target below the horizontal line in Fig. 9. Cases IV-VI denoted by convex curves correspond to the trajectories of the target above the horizontal line in Fig. 9. It is easy to observe that the six cases in Fig. 9 have included all possible affiliations of $d_{vu}$ with the radius of the cluster. Hence, a sensor can join and leave the cluster at most once so the number of messages sent during the probabilistic dynamic clustering before $t_{re}$ is bounded by a constant for each sensor. Note that, the centroid of the cluster is moving while the cluster radius increases. Property 1 eliminates the possibilities of any sensor joining and being removed from the cluster for unbounded times before $t_{re}$. □

Since the message overhead for each sensor is bounded by a constant, to calculate the total message overhead, we just need to calculate how many sensors are included in the cluster at least once during the process. In the case without drift velocity, the cluster with radius $r_s + 2\sqrt{Dt_{re}}$ covers all the sensors.

**Property 2.** *The region traversed by the cluster in the probabilistic dynamic clustering is* convex.

**Proof.** A convex region is a closed region where the line segment connecting any pair of points within the region is also included in the region [35]. We prove the property by calculating the tangent value of angle $\alpha$ in Fig. 11. From time 0 to $t$, the distance traveled by the centroid is equal to $\mu t$, and the increase of cluster radius is $2\sqrt{Dt}$. Therefore, $\tan\alpha$ is equal to the ratio of the radius's increment and the distance traveled by the centroid, $\tan\alpha = 2\sqrt{Dt}/(\mu t) = (2\sqrt{D}/\mu)(1/\sqrt{t})$. Since $\tan\alpha$ is proportional to $1/\sqrt{t}$, $\alpha$ is a monotonically decreasing function. The property of the angle $\alpha$ indicates that the area traversed in probabilistic dynamic clustering is a convex region as shown in Fig. 11. □

Further, for convex region, the lower bound of its area $R$ can be estimated by the combinations of a half of the starting cluster (the blue region), a half of the final cluster (the green region) and the trapezoid connecting them (the orange region). The area of the half of the original cluster is $\pi r_s^2/2$ and final cluster is $\pi(r_s + 2\sqrt{Dt_{re}})^2/2$. The area of the trapezoid is $2(r_s + \sqrt{Dt_{re}})\mu t_{re}$. Therefore, the lower bound of the convex region is,

$$R \geq 2(r_s + \sqrt{Dt_{re}})\mu t_{re} + \frac{\pi[r_s^2 + (r_s + 2\sqrt{Dt_{re}})^2)]}{2}. \quad (23)$$

Note that the probabilistic dynamic clustering has longer $t_{re}$ compared with the original clustering, though the message overhead may increase. Therefore, the lower bound for $R$ is considered to exhibit the potential increment of the message overhead of the probabilistic dynamic clustering.

#### 7.4.4 Comparison of the Message Overhead

Since the density of the sensors is $\rho = n/S$, and the messages sent by each sensor have an upper bound, message overhead of the new clustering strategy is $\mathcal{O}(nR)$ compared to the original clustering strategy that has $\mathcal{O}(n(r_s + 2\sqrt{Dt_{re}})^2)$. If $r_s$ is dominating, then both strategies are $\mathcal{O}(nr_s^2)$. If $t_{re}$ is dominating, then the previous strategy has $\mathcal{O}(nt_{re})$ and the new strategy has $\mathcal{O}(nt_{re}^{3/2})$ which comes from the first term on the right hand side of Eq. (23). If $r_s$ and $t_{re}$ are comparable, then the message overhead of both strategies are $\mathcal{O}(nr_s^2)$. As a result, the probabilistic dynamic clustering algorithm has longer re-clustering time compared with the previous strategy with targets having drift velocity. If re-clustering time is dominating, the new strategy has larger message overhead $\mathcal{O}(nt_{re}^{3/2})$ compared with the message overhead $\mathcal{O}(nt_{re})$ of the old strategy; otherwise, the new strategy achieves the same order of message overhead compared with the old one.

*Remarks:* The precise area $R$ of the convex region can be calculated by applying surface integral (more complex) [36]. This only changes the first term on the right side of the inequality (23) into $(8/3)(r_s + \sqrt{Dt_{re}})\mu t_{re}$ which does not have much difference. The lower bound given by calculating the area of trapezoid and two half clusters has achieved a good estimation in our case.

## 8 Performance Evaluations

We evaluate the performance of the proposed target $k$-coverage WRSN framework by a discrete-event simulator and compare it with previous works that require "all-charge" [7], [8], in which all the zero energy sensors in a cluster are charged by the MC.

In our simulation, $N = 500$ sensors are uniformly randomly distributed in a square sensing field of side length $L = 160$ m and $m = 10$ targets are randomly scattered. Node and target densities are 1.9 nodes/100 $\mathrm{m}^2$, 4 targets/$10^4$ $\mathrm{m}^2$, respectively. Time is equally slotted (1 min) and the average energy consumption rate of working sensor is 12 J/min [38]. A typical sensing range $r_s$ is set to 15 m. Sensors have chargeable Li-Ion battery of 1200 mAh capacity and 3.7 V working voltage with $\Delta t = 30$ mins charging time from empty to full. The MC moves at a constant speed of 1 m/s [39] and consumes $e_t = 5$ J/m. When the percentage of sensors that can work in a cluster is lower than a threshold $\eta = 20\%$, the cluster head sends out an energy request. The total simulation time is typically set to 60 days. For all of the following simulation results, the standard deviation for each data point is derived through 100 times independent experiments for each case, which are shown as the error bars in each figure.

### 8.1 Charging Capability

First, we evaluate charging capability of MC in the new framework compared with previous works of all-charge [7], [8]. The charging capability is measured by the maximum area covered by one MC without violating $k$-coverage.

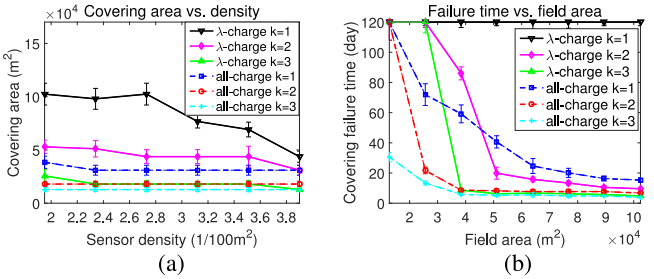



Fig. 12. Covering area and covering failure time. (a) Covering area of one MC versus sensor density. (b) Target $k$-covering failure time versus field area.

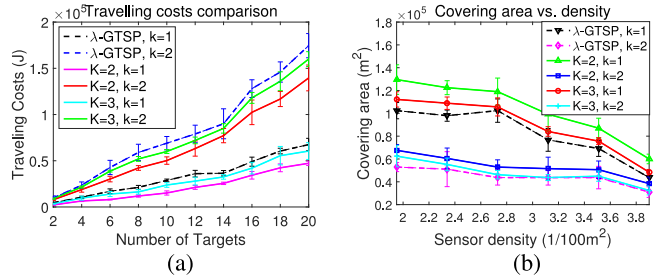



Fig. 14. Traveling cost and covering area of MC by comparing $\lambda$-GTSP and $K$-means working sensor determination algorithm. (a) Moving cost versus number of targets. (b) Covering area of one MC versus sensor density.

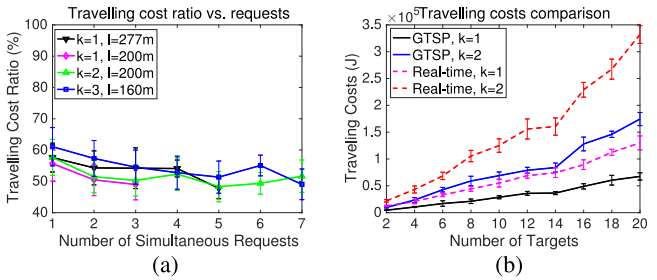



Fig. 13. Comparison of moving cost of MC. (a) Moving cost ratio versus number of simultaneous energy requests. (b) Moving cost versus number of targets.

In Fig. 12, our algorithm is called $\lambda$-charge. For an area with lentgh $L$, two schemes must succeed 100 independent tests in order to cover it.

Fig. 12a compares covering area of MC between charging $\lambda_i$ and all the nodes in a cluster, where $k$ is varied from 1-3 for both cases. For fairness in comparison, "all-charge" algorithm also just needs to maintain target $k$-coverage instead of target all-coverage during simulation. First, we observe that the covering area decreases with the increment of $k$ for both frameworks. This is because higher $k$ means higher energy consumptions and energy demands, which confines the charging scope of MC. Second, for specific $k$-coverage requirement, our framework surpasses the previous framework on covering area of MC. Our framework still achieves about 50 percent increase of covering area of MC for $k = 1, 2$ even if node density is doubled.

Next, to examine the performance of our framework further, we allow the simulation to run longer until $k$-coverage no longer holds, as shown in Fig. 12b. Reaching 120 days means that the MC maintains $k$-coverage over the entire time period. First, failure occurs much faster in a larger field, since larger field incurs higher energy cost and MC can barely satisfy all the energy demands. Second, for the same field , our framework can support the network much longer than the all-charge framework.

It is worth mentioning that when $k = 1$, our framework with only one MC successfully accomplishes $k$-coverage at any time during the 120 days whereas the curve of all-charge framework drops sharply which can only sustain 15 days. The results show that our new framework can extend network lifetime by a large extent.

## 8.2 Moving Cost of MC

We compare the moving cost in our framework (applying $\lambda$-GTSP) to the real-time charging algorithm proposed in [8]. For fairness, the algorithm in [8] also charges the same number of $\lambda_i$ sensors in each cluster whereas the locations of these sensors are picked randomly and the choices of charged clusters are planned at real-time.

Fig. 13a shows the ratio of the moving cost of MC using our $\lambda$-GTSP algorithm to the cost of the real-time algorithm. We can see that our algorithm saves 40%-50% energy due to $\lambda$-GTSP charging algorithm and careful selection of $\lambda_i$ sensors in each cluster. It is interesting to see that receiving more simultaneous charging requests actually helps us reduce more operating cost on the MC compared to the scheme in [8] that only charges the next nearest cluster. This is because the MC always enjoys the benefits brought by $\lambda$-GTSP once a charging route is planned appropriately.

Fig. 13b shows the relation between the total traveling cost of MC and the number of targets. The traveling cost of MC is defined as the energy consumed by MC for traveling in the whole simulation duration (e.g., 120 days). The cost increases linearly as the number of targets grows. This is because that MC has to meet rising energy demands from sensors (monitor more targets). For different $k$-coverage requirements, $\lambda$-GTSP can always provide a cost saving of more than 50 percent as the number of targets increases.

## 8.3 Charging Capability and Moving Cost of the MC

In this section, the covering capability and traveling cost of MC are evaluated by comparing the algorithm proposed in Section 6.6 with the $\lambda - GTSP$ algorithm that randomly picks $k$ working sensors. The original algorithm is compared with the case where $K = 2$ and $K = 3$. For the same $k$ value, the traveling cost decreases if $k$ sensors are chosen wisely via the $K$-means partition. $K = 2$ gives the best cost saving on the MC among different $K$ values. For $k = 1$, $K = 2$ and $K = 3$ increase the covering area of one MC by an average of 35 and 14 percent respectively. If $K \geq 3$, the size of each sub-cluster decreases. New sub-clusters need to be picked frequently, which may potentially increase MC's traveling cost. Thus, $K = 2$ yields the best saving. In Fig. 14b, we can see that covering area of MC shows similar results. This is because energy saving on MC provides more energy for charging and better schedules to meet sensors' battery deadlines.

## 8.4 Coverage Percentage and Re-Clustering Time for Mobile Targets

Next, we evaluate the performance of our framework for mobile targets. Fig. 15a shows the average percentage of targets being $k$-covered. 15 targets are randomly distributed in a square field of side length 160 m. 2000 sensors are distributed in a larger square field of side length 320 m containing the smaller field, which alleviates the impact of field boundaries. The diffusion coefficient of target $D$ is set to

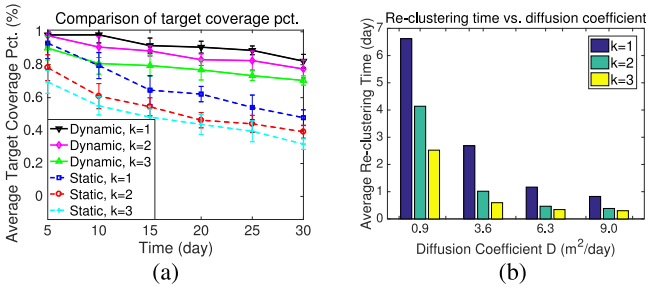



Fig. 15. Average target $k$-coverage percentage and re-clustering time. (a) Comparison of target $k$-coverage percentage between dynamic clustering and static clustering. (b) Re-clustering time versus diffusion coefficient for different $k$.

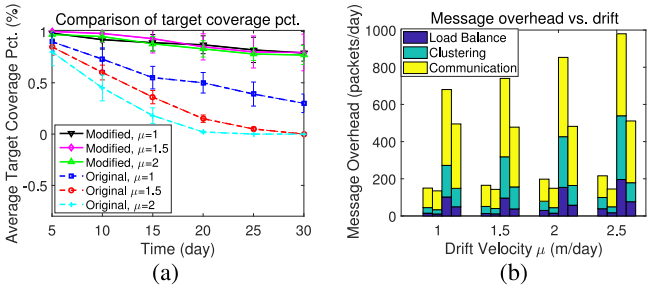



Fig. 16. Average target 1-coverage percentage and message overhead. (a) Comparison of target 1-coverage percentage for original clustering strategy and modified clustering strategy. (b) Stacked message overhead (for the same $\mu$, from the left to the right are D = 2 modified, D = 2 original, D = 4 modified, and D = 4 original, respectively) versus drift velocity $\mu$ for different diffusion coefficient $D(\mathrm{m}^2/\mathrm{day})$.

$3.6\ \mathrm{m}^2/\mathrm{day}$. The target coverage rate is the ratio between $k$-covered and total targets. For the same $k$ value, dynamic clustering can maintain much higher target coverage rate compared to the static one. Coverage declines with time, since the variance of target locations is getting larger. For $k = 1$-3, dynamic clustering can maintain at least 80 percent coverage rate in 30 days.

Fig. 15b demonstrates the change of average re-clustering time for different target mobility. If a target moves out of the cluster or the MC can no longer fulfill the charging requests, re-clustering is needed. As shown, the re-clustering time decreases as the diffusion coefficient of target increases and smaller $k$ corresponds to longer re-clustering time. As for the re-clustering frequency, it increases as the diffusion coefficient of target increases and smaller $k$ corresponds to smaller re-clustering frequency.

### 8.5 Coverage Percentage and Re-Clustering Time

We evaluate the target coverage percentage in time by comparing the proposed strategies. The results are shown in Fig. 16a. The movement of targets follows Brownian motion with drift. The diffusion and drift coefficients are $3.6\ \mathrm{m}^2/\mathrm{day}$ and $2\ \mathrm{m}/\mathrm{day}$ respectively [40]. Target is successfully covered if it is 1-covered.

The drift velocity $\mu$ is set in the range from 1 m/day to 2.5 m/day and the diffusion coefficient is set in the range from $0.9\ \mathrm{m}^2/\mathrm{day}$ to $9.0\ \mathrm{m}^2/\mathrm{day}$. At the current stage, our WSN is designed to monitor targets with slow motions such as glacier melting, spread of desert, or herds that move slowly. For higher speed targets, a solution is to form clusters according to the current positions of targets. Yet, it suffers from much higher message overhead to keep track of the targets and the subsequent charging strategy will be also affected.

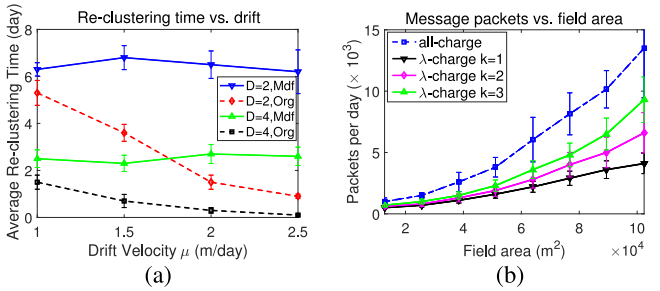



Fig. 17. Average re-clustering time and the packets of the whole network. (a) Average re-clustering time versus the drift velocity $\mu$ of the mobile targets. (b) Packets of the whole network versus the area of the field.

In Fig. 16a, it is observed that the average target coverage percentages for both strategies decrease in time. However, the modified strategy exhibits much higher target coverage rate for all the $\mu$. It is also observed that the target coverage rate of modified strategy is almost the same for three $\mu$ values. This is reasonable since the modified strategy adjusts the positions of the cluster's centroid according to the expected position of the target. In contrast, the original strategy is affected by the $\mu$ value. The slope of the coverage rate is decreasing rapidly with the increase of $\mu$ values. The target tends to get closer to the boundary or even moves out of the cluster, resulting in failed coverage.

In Fig. 16b, we evaluate the message overhead for both clustering strategies. Note that, We adopt CSMA/CA protocol for the MAC layer and handle packet loss by ARQ (Automatic Repeat reQuest) [41]. The packet loss and retransmission scheme are accumulated into the message overhead in our simulation.

The message overhead is shown in a stacked manner. Threes components of the message overhead includes the clustering/re-clustering overhead, load balance overhead and communication overhead within clusters. Fig. 16b shows that the communication contributes to the most part, while the clustering contributes the next and load balance contributes to the least. The increase of message overhead is compensated by the decrease of re-clustering overhead.

### 8.6 Message Overhead and Total Packets

In Fig. 17a, we evaluate the average re-clustering time of different clustering strategies for the mobile targets. For same $D$, the re-clustering time is not affected by $\mu$ for the modified strategy. In contrast, re-clustering time drops rapidly for the original strategy. For the same strategy, the increase of diffusion coefficient results in the decrease of re-clustering time. It is obvious that the modified strategy has significant increase of re-clustering time compared to the original strategy and demonstrates good performance with an increasing $\mu$.

In Fig. 17b, the packets transmitted or received through the whole network is simulated and are shown in the units of kilo packets per day. The total packets increase faster with the increasing of the field area, since larger field involves more communications and needs to spend more energy to transmit the messages to the destination considering the distance and possible failures. Smaller $k$ corresponds to fewer packets since fewer sensors are in the working mode and fewer data need to be transmitted.

## 9 Conclusions

In this paper, we have considered target $k$-coverage in WRSNs. First, we conduct theoretical analysis on the improvement of charging capability of MC by only charging a portion of sensors. Second, we study a distributed algorithm that can assign sensors into balanced clusters around targets. Third, we optimize the number of sensors being charged in each cluster while guaranteeing target $k$-coverage. A $\lambda$-GTSP charging algorithm is proposed while working sensors are picked wisely. Next, we further consider mobile targets of different motion patterns such that original clusters are expanded until a re-clustering condition is met. Finally, we demonstrate that the new framework can greatly improve the charging capability of MC and reduce the operating cost.

## Acknowledgments

The work in this paper was supported by the U.S. National Science Foundation under Grant No. ECCS-1307576.

## References

[1] S. He, J. Chen, F. Jiang, D. Yau, G. Xing, and Y. Sun, "Energy provisioning in wireless rechargeable sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 10, pp. 1931–1942, Oct. 2013.

[2] H. Dai, Y. Liu, G. Chen, X. Wu, and T. He, "Safe charging for wireless power transfer," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 1105–1113.

[3] M. Ma and Y. Yang, "SenCar: An energy efficient data gathering mechanism for large scale multihop sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 10, pp. 1476–1488, Oct. 2007.

[4] C. Wang, J. Li, F. Ye, and Y. Yang, "Improve charging capability for wireless rechargeable sensor networks using resonant repeaters," *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, 2015, pp. 133–142.

[5] C. Wang, J. Li, Y. Yang, and F. Ye, "A hybrid framework combining solar energy harvesting and wireless charging for wireless sensor networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[6] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Prolonging sensor network lifetime through wireless charging," in *Proc. 31st IEEE Real-Time Syst. Symp.*, 2010, pp. 129–139.

[7] Y. Shi, L. Xie, Y. Hou, and H. Sherali, "On renewable sensor networks with wireless energy transfer," in *Proc. IEEE INFOCOM*, 2011, pp. 1350–1358.

[8] C. Wang, J. Li, F. Ye, and Y. Yang, "NETWRAP: An NDN based real-time wireless recharging framework for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1283–1297, Jun. 2014.

[9] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor netwokrs," in *Proc. IEEE 3rd Annu. Commun. Netw. Services Res. Conf.*, 2005, pp. 255–260.

[10] L. Qing, Q. Zhu, and M. Wang, "Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks," *Comput. Commun.*, vol. 29, pp. 2230–2237, 2006.

[11] M. Zhao, J. Li, and Y. Yang, "A framework of joint mobile energy replenishment and data gathering in wireless rechargeable sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 12, pp. 2689–2705, Dec. 2014.

[12] C. Huang and Y. Tseng, "The coverage problem in a wireless sensor network," in *Proc. ACM Int. Conf. Wireless Sensor Netw. Appl.*, 2003, pp. 115–121.

[13] F. Li, J. Luo, S. Q. Xin, W. P. Wang, and Y. He, "LAACAD: Load balancing k-area coverage through autonomous deployment in wireless sensor networks," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 566–575.

[14] F. Li, J. Luo, S. Q. Xin, W. P. Wang, and Y. He, "Autonomous deployment for load balancing $k$-surface coverage in sensor networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 279–293, Jan. 2015.

[15] K. Ioannis and S. Shreve, *Brownian Motion and Stochastic Calculus*, Berlin, Germany: Springer, 2012.

[16] W. Wei, T. He, C. Bisdikian, D. Goeckel, B. Jiang, L. Kaplan, and D. Towsley, "Impact of in-network aggregation on target tracking quality under network delays," *IEEE J. Selected Areas Commun.*, vol. 31, no. 4, pp. 808–818, Apr. 2013.

[17] N. Baccour, A. Koubaa, L. Mottola, M. Zuniga, H. Youssef, C. Alberto Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," ACM *Trans. Sensor Netw.*, vol. 8, no. 4, 2012, Art. no. 34.

[18] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, Art. no. 10.

[19] V. Dimitrijevic and Z. Saric, "An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs," *Inf. Sci.*, vol. 102, pp. 105–110, 1997.

[20] P. Jaillet, "Probabilistic Traveling Salesman Problem," Ph.D. Dissertation, Department of EECS, MIT, Cambridge, MA, 1985.

[21] A. Kurs, A. Karalis, R. Moffatt, J. Joannopoulos, P. Fisher, and M. Soljacic, "Wireless power transfer via strongly coupled magnetic resonances," *Sci.*, vol. 317, no. 5834, pp. 83–86, 2007.

[22] [Online]. Available: http://www.afar.net/tutorials/fcc-rules, *FCC Rules for Unlicensed Wireless Equipment Operating in the ISM Bands*, 2018.

[23] Q. Zhao and M. Gurusamy, "Lifetime maximization for connected target coverage in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1378–1391, Dec. 2008.

[24] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor newworks," in *Proc. IEEE 24th Annu. Joint Conf. Comput. Commun. Societies.*, 2005, pp. 1976–1984.

[25] J. Ai and A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *J. Combinatorial Optimization*, vol. 11, no. 1, pp. 21–41, 2006.

[26] D. Hall and S. McMullen, *Math. Techn. Multisensor Data Fusion*. Norwood, MA, USA: Artech House, 1992.

[27] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1998.

[28] IBM CPLEX, *User's Manual for CPLEX*. Armonk, NY, USA: International Business Machines Corporation, 2009.

[29] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–97.

[30] D. M. Christopher, P. Raghavan and S. Hinrich "Introduction to information retrieval," in *An Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[31] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, pp. 231–247, 1992.

[32] K. E. McCluney and J. L. Sabo, "Tracing water sources of terrestrial animal populations with stable isotopes: Laboratory tests with crickets and spiders," *PloS One*, vol. 5, no. 12, 2010, Art. no. e15696.

[33] (2017). [Online]. Available: https://www.nps.gov/fire/wildland-fire/learning-center/fire-in-depth.cfm, Fire In-Depth.

[34] A. N. Borodin, and S. Paavo, "Brownian motion with drift," in *Handbook of Brownian MotionFacts and Formulae*. Berlin, Germany: Springer, 1996.

[35] (2018). [Online]. Available: https://en.wikipedia.org/wiki/Convex_set, *Convex Set, Wikipedia*.

[36] (2018). [Online]. Available: https://en.wikipedia.org/wiki/Surface_integral, *Surface Integral, Wikipedia*.

[37] B. Tong, G. Wang, W. Zhang, and C. Wang, "Node reclamation and replacement for long-lived sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1550–1563, Sep. 2011.

[38] S. Chen, J. Yao, and Y. Wu, "Analysis of the power consumption for wireless sensor network node based on Zigbee," *Procedia Eng.*, vol. 29, pp. 1994–1998, 2012.

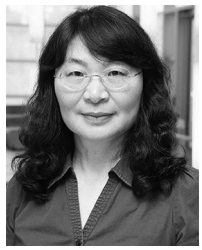
[39] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Prolonging sensor network lifetime through wireless charging," in *Proc. 31st IEEE Real-Time Syst. Symp.*, 2010, pp. 129–139.

[40] B. McClintock, R. King, L. Thomas, J. Matthiopoulos, B. McConnell, and J. Morales, "A general discret-time modeling framework for animal movement using multistate random walks," *Ecological Monographs*, vol. 82, no. 3, pp. 335–349, 2012.

[41] S. Khan, M. Moosa, F. Naeem, M. Alizai, and J. Kim, "Protocols and mechanisms to recover failed packets in wireless networks: History and evolution," *IEEE Access*, vol. 4, pp. 4207–4224, 2016.

**Pengzhan Zhou** received the BS degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China. He is currently working towards the PhD degree in the Department of Electrical and Computer Engineering, Stony Brook University, New York. His research interests include wireless sensor networks and performance evaluation of network protocols and algorithms.

**Cong Wang** received the BEng degree in information engineering from the Chinese University of Hong Kong, and the PhD degree in computer and electrical engineering from Stony Brook University. He is an assistant professor with Old Dominion University in Norfolk, VA. His research interests include mobile computing, data privacy, energy efficiency, and optimization.



**Yuanyuan Yang** received the BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and the MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a SUNY distinguished professor of computer engineering and computer science at Stony Brook University, New York, and is currently on leave at the National Science Foundation as a program director. Her research interests include edge computing, data center networks, cloud computing, and wireless networks. She has published more than 400 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the associate editor-in-chief for the *IEEE Transactions on Cloud Computing* and an associate editor for *ACM Computing Surveys*. She has served as an associate editor-in-chief and associate editor for the *IEEE Transactions on Computers* and associate editor for the *IEEE Transactions on Parallel and Distributed Systems*. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is a fellow of the IEEE.