Leveraging Target k-Coverage in Wireless Rechargeable Sensor Networks

Pengzhan Zhou, Cong Wang and Yuanyuan Yang
Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794, USA

***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 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 λ-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% of moving cost without sacrificing the network performance.**

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

I. Introduction and Related Work

Recent advances in technology are leading the trend to launch intelligent applications on smart phones, wireless sensors and wearable devices, and integrate them into the Internet-of-Things. These applications usually rely on ubiquitous 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]–[7], [9], [10], [13].

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], [8]–[10], [13]. 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 wireless rechargeable sensor networks. In [5], a hybrid network consisting of solar harvesting sensors, wireless rechargeable sensors and mobile chargers are proposed to reduce the energy consumption of the network while maintaining the performance. In [8], a greedy algorithm is designed to find a charging sequence to maximize network lifetime. In [9], a shortest Hamilton cycle is pre-planned through all the sensors for wireless charging. In [10], MC receives real-time energy status from sensors and makes charging decisions on-the-fly. In [13], joint wireless charging and mobile data gathering is considered. For high charging efficiency, in this paper, we will focus on inductive wireless charging.

Most of 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.

It is observed that due to low cost of wireless sensors, they are usually deployed at high density so there is abundant redundancy in their coverage in the network. 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 [14], where k is a user-input based on various task requirements. In fact, with advances in pattern recognition and 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 functionality. 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?

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 to 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 λ-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 which characterizes when the process should terminate and a new round of clustering should start.

The contributions of this paper can be 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 previous works which only focus on static targets, we extend our work to cover mobile targets as well. Finally, we conduct extensive simulations to evaluate the performance of the new framework and compare with previous works [8]–[10] under all-charge and real-time charging strategies. Our results indicate that the new framework can extend charging capability of MC significantly (over 3 times of covering area) and reduce about 40% of operating cost of MC. Meanwhile, more than 80% target coverage rate is maintained for mobile targets during operation.

The rest of the paper is organized as follows. Section II introduces the network model, assumptions and motivations of our work. Section III theoretically compares the charging capability of MC based on different charging strategies. Section IV studies how to balance the cluster size. Section V derives the number of sensors to be charged in each cluster and plans the charging route for MC. Section VI further considers mobile targets. Section VII evaluates the performance of the new framework and Section VIII concludes the paper.

TABLE I
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
η	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
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
Δt	Time to charge a zero energy sensor to full capacity
λ_i	Number of sensors to be charged in cluster i
t_{re}	Time for re-clustering for mobile targets

II. PRELIMINARIES

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

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.

We first analyze the case when targets are static [19]–[21]. In practice, many applications require sensors to monitor static targets such as security surveillance, traffic monitoring, etc. After solving the static target case, we will also consider the targets following a random mobility pattern.

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. In the case that a sensor can detect one or more targets within r_s (i.e., can join multiple clusters), we will give an algorithm in Section IV to resolve such contention and ensure the size of neighboring clusters is balanced.

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 [22]. 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, which reduces to a 1-coverage problem. On the other hand,

k can also be extended to n_i, i.e., all the sensors in a cluster are in working mode to provide full-coverage, which has been the predominant method in previous works [1], [2], [9], [10]. 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 average lifetime l. During operating, exact k sensors are in working mode, while others remain in the 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 η 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 Δt. For k-coverage, MC charges λ_i nodes in cluster C_i, where the value of λ_i is to be optimized in Section V-C. 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.

Since we only charge a portion of sensors, λ_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) [17]. 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 λ_i nodes in cluster C_i. Hence, we call our new problem *λ-GTSP*.

The question is whether λ-GTSP can help us reduce the moving cost of MC. The full-coverage in [9], [10] 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 $\sqrt{2(n-2)D_1D_2} + 2(D_1 + D_2)$ [18]. In contrast, in k-coverage, we only charge λ_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 tour length of MC derived by λ-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.

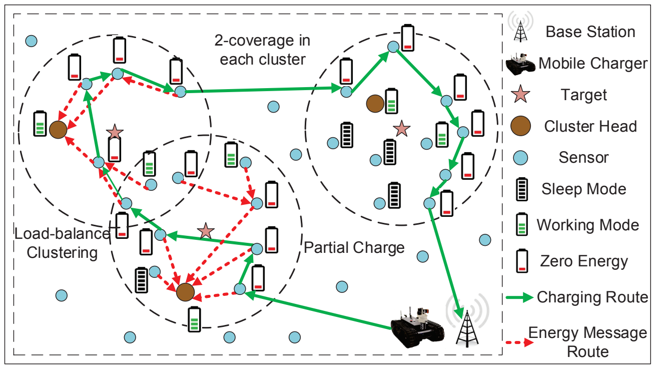


Fig. 1. An example of λ-GTSP charging framework providing target 2-coverage for 3 targets.

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

III. 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 [9], [10], 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 VII.

A. *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 η percentage of alive nodes, it sends out

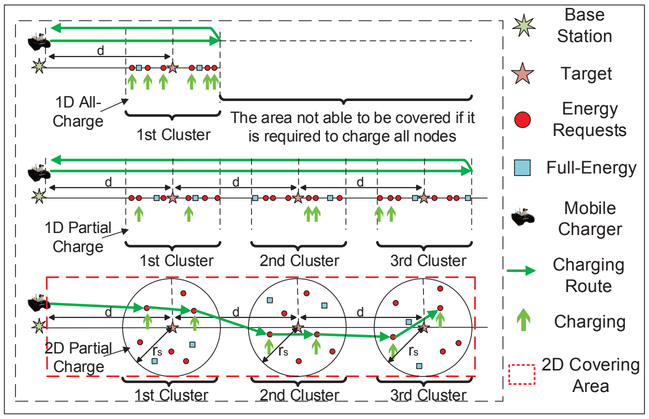


Fig. 2. Comparison of the scope covered by all-charge and partial-charge.

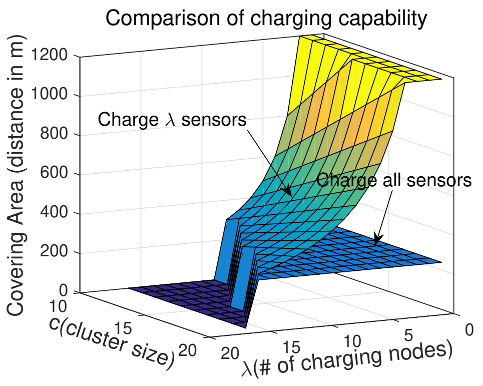


Fig. 3. Comparison of charging capability of MC by all-charge and partial-charge. $d = 60$m, $l = 20$h, $\Delta t = 0.5$h, $\eta = 20\%$, $k = 2$.

a charging request. The lifetime of a sensor is l, the charging time from zero to full capacity is Δt, 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 λ nodes in each cluster. In fact, λ could be different for different clusters based on their energy status and we will further optimize the value of λ in Section V. We compare the covering distance of the two approaches.

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 ηc 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 η 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 ϵ, so the traveling time can be ignored. Note that ϵ 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 $x \le \frac{\eta l}{(1-\eta)k\Delta t}$.

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

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

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 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 λ nodes scales much better than charging all nodes. For fixed cluster size c, the two approaches converge as λ increases whereas more benefits are brought by charging only a small λ number of nodes, e.g., the improvements can be as high as 4 times.

B. 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 II, 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 $2r_s(\sqrt{2c}+2) \le d$, which is

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

By applying Eq. (3) and Eq. (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 ρ_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 III-A.

IV. 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.

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 [11], a centralized algorithm is proposed to realize real-time head selection based on node concentration, energy level and centrality. In [12], 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 [12] 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 target 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 ω_l. $|\omega_l|$ is the tuple size equal to the number of targets contained in ω_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 ω_l into a set Ψ_l. All these different Ψ_l form a bigger set $\{\Psi_l\}$. Next, we first give an example for handling tuples with $|\omega_l| = 2$.

TABLE II
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 $u \in \Psi_l$;
    $k \leftarrow \arg\min_{i \in \omega_l}\{n_i\}$;
    $C_k \leftarrow C_k \bigcup \{u\}$; $n_k \leftarrow n_k + 1$;
    $\Psi_l = \Psi_l \setminus \{u\}$.
  end while
end for
```

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 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 Ψ_1. Among the targets in tuple ω_1, we find target T_k corresponding to the cluster C_k with minimum n_k. A node u is randomly picked from Ψ_1, and assigned to cluster C_k. Thus, n_k is increased by 1. Then we remove node u from Ψ_1. For the remaining nodes in Ψ_1, we repeat the above process until Ψ_1 is empty. The iteration continues for Ψ_2, Ψ_3, ..., until the set $\{\Psi_l\}$ is exhausted. This algorithm is summarized in Table II.

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

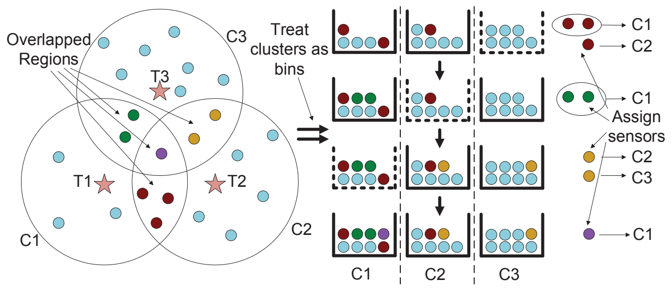


Fig. 4. An example of balancing nodes in neighboring clusters.

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 number of iterations is at most n, so the overall message overhead is $\mathcal{O}(n)$.

V. 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 λ-GTSP charging route, and give an example to demonstrate the complete process in this section. The above steps are altogether summarized as the λ-GTSP Charging Algorithm.

A. 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 λ_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 $d'_{uv} = d_{uv}$, 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.

B. Transforming GTSP to TSP

Recall the definition in Section II, 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 V-C, V-D, 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 [17]. 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 counter-clockwise), 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.

C. Optimizing λ_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,

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

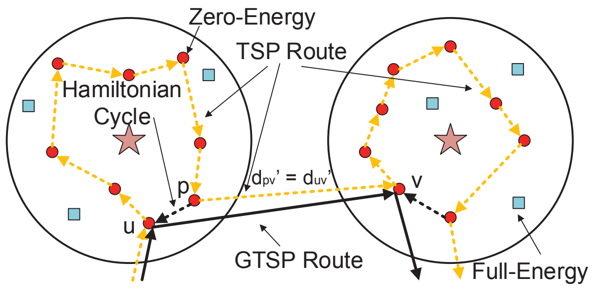


Fig. 5. Derivation of GTSP route by solving TSP.

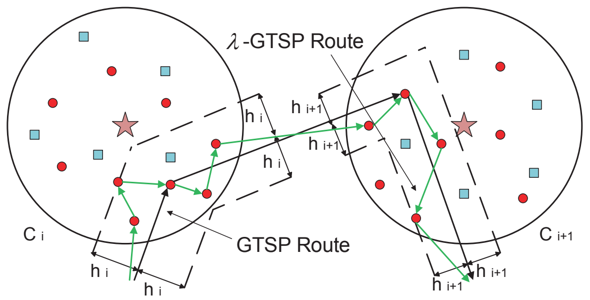


Fig. 6. Derivation of λ-GTSP charging route.

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. λ_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 less 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 [23], 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 λ_i [24].

D. Calculating λ-GTSP

After λ_i has been calculated for each cluster, MC selects which λ_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 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 λ_i nodes and TSP is solved in each cluster to connect λ_i picked nodes by a shortest path. Finally, a λ-GTSP charging route is generated which traverses through λ_i nodes in each cluster calculated by the IP in Eq. (10).

An example of the λ-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 Program **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 δ each time until λ_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 λ_i nodes in C_i is the λ-GTSP route represented by the green arrows in Fig. 6.

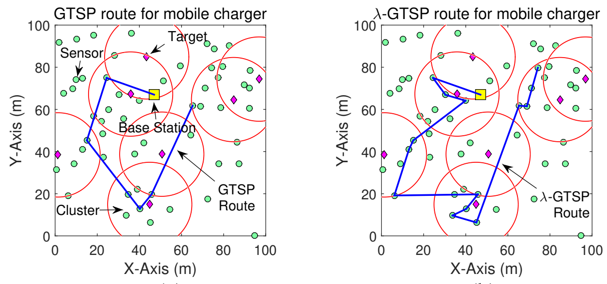
The above calculations can be done at the base station and disseminated to MC through long range wireless communications such as LTE. We summarize all the above process discussed in section V as λ-GTSP Charging Algorithm in Table III.

TABLE III
λ-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, $d'_{pu} \leftarrow 0$;
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 $\leftarrow \bigcup_i \{$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
  $h_i \leftarrow h_i + \delta$, construct sweeping sector of width $h_i$;
  $Z_i \leftarrow$ 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 $\bigcup_i \{Z_i\}$.
```

E. An Example of λ-GTSP Network

We demonstrate the process of λ-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 IV. Upon receiving charging requests from the clusters, base station derives the GTSP route denoted as the blue lines in Fig. 7(a) based on the new distance metric. Finally, the λ-GTSP



(a) (b)

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

algorithm calculates number of charged nodes λ_i for each cluster and iteratively finds those sensors along the original GTSP trajectory and the corresponding λ-GTSP route in Fig. 7(b).

VI. Extension to 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.

A. *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 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*,

$$<x^2> = 2 \cdot \mathit{dim} \cdot Dt, \tag{13}$$

where *dim* is the dimensions of the space (2 for the 2-dimensional sensing field), and D (m^2/s) represents the diffusion coefficient of target, 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 it the transmission range of sensor. 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.

B. *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 πr_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 III, if the target density ρ_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 $\mathcal{A}$.

$$\frac{l r_s^2}{(r_s + 2\sqrt{Dt})^2 k \Delta t} \geq m. \tag{14}$$

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

$$t_{re} = \min\left\{\tau, \frac{r_s^2}{4D}\left(\sqrt{\frac{l}{km\Delta t}} - 1\right)^2\right\}, \tag{15}$$

where τ is the time when the target moves out of cluster; otherwise, $\tau = \infty$.

VII. 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" [9], [10], 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/100m^2, 4 targets/10^4m^2, respectively. Time is equally slotted (1 min) and the average energy consumption rate of working sensor is 12 J/min. 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 10 m/min 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.

A. Charging Capability

First, we evaluate charging capability of MC in the new framework compared with previous works of all-charge [9], [10]. The charging capability is measured by the area that one MC can cover without violating target k-coverage at any time during the operation. In Fig. 8, our algorithm is called λ-charge since only λ_i sensors are charged in each cluster. For a given field length L, we test the two schemes 100 times with a new random distribution of sensors and targets each time. If any iteration fails to satisfy k-coverage, we state that MC can no longer cover the field of length L.

Fig. 8(a) compares covering area of MC between charging λ_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 regardless of which framework is considered, the covering area decreases with the increment of k. This is because that when k increases, more sensors are turned into working mode thus higher energy consumptions and energy demands are observed. Clusters would have energy requests more frequently, which confines the charging scope of MC. Second, for specific k-coverage requirement, our framework surpasses the previous framework on covering area of MC. For example, when $k = 1$ and low node density (less than 2.7 nodes/100m^2), our algorithm grows the covering area by an average of 3 times and $2.4, 1.5$ times for $k = 2$, 3, respectively. Although a higher node density slightly diminishes the benefits of k-coverage framework (due to higher energy consumptions), our framework still achieves about 50% 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. The exact time of such failure is plotted versus the increase of sensing field in Fig. 8(b). The failure time is averaged over 100 simulation runs. We set the upper limit to be 120 days just in case the MC successfully maintains k-coverage over the entire time period. Note that when we increase the field length, node and target densities are retained (1.9 nodes/100m^2, 4 targets/10^{4}m^2). We first observe that failure occurs much faster with a larger sensing field. This is intuitive since larger field with more targets incurs much higher energy cost and MC can barely satisfy all the energy demands. Second, for the same field size, 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. For

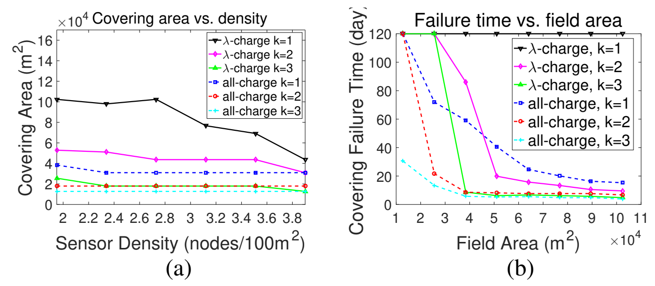


Fig. 8. Covering area and covering failure time. (a) Covering area of one MC vs. sensor density. (b) Target k-covering failure time vs. field area.

the failure time, the results demonstrate our framework can support the network much longer. For example, when $k = 2$, our framework lasts for 9.6 days and the all-charge framework only runs for 6.8 days with the field size of 1.0×10^5 m^2. When $k = 2$, the network runs for 20 days with the field size of 5×10^4 m^2 while the same 20-day operation requires the all-charge framework with less than 2.5×10^4 m^2 field size (a complete half of the area size). When $k = 3$ with a field size less than 2.6×10^4 m^2, our framework can still last for more than 120 days while the all-charge network only runs for 13 days. In sum, the results show that our new framework can extend network lifetime by a large extent.

B. Moving Cost of MC

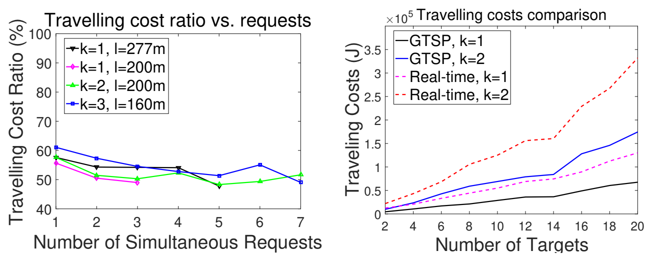
Second, we compare the moving cost in our framework (which applies λ-GTSP) to the real-time charging algorithm proposed in [10]. For fairness, the algorithm in [10] also charges the same number of λ_i sensors in each cluster whereas the locations of these sensors are picked randomly and the choices of which clusters need charging are planned at real-time.

Fig. 9(a) shows the ratio of moving cost of MC using our λ-GTSP algorithm to the cost of real-time algorithm. We can see that our algorithm saves $40\% - 50\%$ energy due to λ-GTSP charging algorithm and careful selection of λ_i sensors in each cluster. We examine the evolution of this ratio with respect to the number of simultaneous charging requests. 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 [10] that only charges the next nearest cluster. This is because the MC always enjoys the benefits brought by λ-GTSP once a charging route is planned appropriately.

Fig. 9(b) shows the relation between the total moving cost of MC and number of targets. 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, λ-GTSP can always provide a cost saving of more than 50% as the number of targets increases. We also notice that $k = 2$ is larger than $k = 1$. It verifies that a higher k would incur higher operating cost on MC.

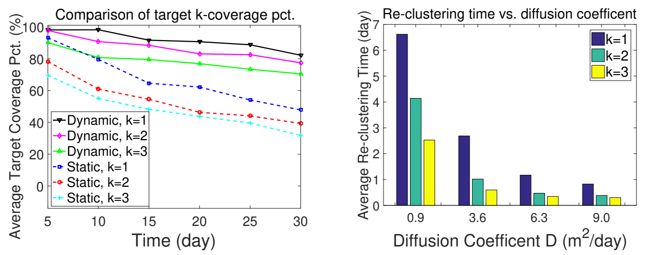
C. Coverage Percentage and Re-clustering Time for Mobile Targets

Finally, we evaluate the performance of our framework for mobile targets. Fig. 10(a) compares the average percentage of targets being k-covered by our dynamic clustering algorithm



(a) (b)

Fig. 9. Comparison of moving cost of MC. (a) Moving cost ratio vs. number of simultaneous energy requests. (b) Moving cost vs. number of targets.



(a) (b)

Fig. 10. Average target k-coverage percentage and re-clustering time. (a) Comparison of target k-coverage percentage for dynamic clustering and static clustering. (b) Re-clustering time vs. diffusion coefficient for different k.

with the static clustering algorithm summarized in Table II. In the simulation, 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 which contains the smaller field. The diffusion coefficient of target D is set to 3.6 m^2/day. The target coverage rate is the ratio of the number of targets being k-covered to the total number of targets, which is sampled every 1 min. The average target coverage percentage shown in the figure is the average target coverage rate over every 5 days. The coverage percentages are derived from 100 simulation runs.

For the same k value, the dynamic clustering scheme can maintain much higher average target coverage percentage compared to static clustering. The coverage percentages decrease with time, since the variance of target locations is getting larger. Meanwhile, a smaller k corresponds to higher coverage percentage, since it is easier to satisfy the coverage requirement. For $k = 1, 2, 3$, our dynamic clustering scheme can maintain at least 80% coverage rate within 30 days.

Fig. 10(b) demonstrates the change of average re-clustering time for different target mobility. According to our approach, if any target moves out of the cluster or the MC can no longer fulfill the charging requests of all clusters, re-clustering is needed. The re-clustering time is averaged during 120 days simulation time and over 100 simulation runs. As shown in the figure, the average re-clustering time decreases as the diffusion coefficient of target increases and smaller k corresponds to longer re-clustering time.

VIII. 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 λ-GTSP charging algorithm is proposed. Next, we further consider mobile targets 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.

IX. Acknowledgment

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 Computing*, 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," *IEEE INFOCOM*, 2014.
[3] M. Ma and Y. Yang, "SenCar: An energy efficient data gathering mechanism for large scale multihop sensor networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1476-1488, 2007.
[4] C. Wang, J. Li, F. Ye, and Y. Yang, "Improve charging capability for wireless rechargeable sensor networks using resonant repeaters," *IEEE ICDCS*, 2015.
[5] C. Wang, J. Li, Y. Yang and F. Ye, "A hybrid framework combining solar energy harvesting and wireless charging for wireless sensor networks," *IEEE INFOCOM*, 2016.
[6] S. Nikoletseas, Y. Yang and A. Georgiadis, *Wireless Power Transfer Algorithms, Technologies and Applications in Ad Hoc Communication Networks*, Springer, 2016. 978-3-319-46809-9.
[7] Y. Yang and C. Wang, *Wireless Rechargeable Sensor Networks,* Springer, 2015.
[8] Y. Peng, Z. Li, W. Zhang and D. Qiao, "Prolonging sensor network lifetime through wireless charging," *IEEE RTSS*, 2010.
[9] Y. Shi, L. Xie, Y. Hou and H. Sherali, "On renewable sensor networks with wireless energy transfer," *IEEE INFOCOM*, 2011.
[10] 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 Computing*, vol. 13, no. 6, pp. 1283-1297, 2014.
[11] I. Gupta, D. Riordan and S. Sampalli, "Cluster-head election using fuzzy logic for wireless sensor netwokrs," *IEEE CNSR*, 2005.
[12] L. Qing, Q.Zhu and M. Wang, "Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks," *Computer Communications*, vol. 29, pp. 2230-2237, 2006.
[13] 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 Computing*, vol. 13, no. 12, pp. 1536-1233, 2014.
[14] C. Huang and Y. Tseng, "The coverage problem in a wireless sensor network," *Proc. ACM Int. Conf. Wireless Sensor Networks and Application(WSNA)*, pp. 115-121, 2003.
[15] K. Ioannis and S. Shreve, "Brownian motion and stochastic calculus," *Springer Science and Business Media*, 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 JSAC*, 2013.
[17] V. Dimitrijevic and Z. Saric, "An efficient transformation of the generalized traveling salesman problem into the traveling salesman problem on digraphs," *Information Science*, 1997.
[18] P. Jaillet, *Probabilistic Traveling Salesman Problem*, Ph.D. Dissertation, MIT, 1985.
[19] Q. Zhao and M. Gurusamy, "Lifetime maximization for connected target coverage in wireless sensor networks," *IEEE/ACM Trans. Networking*, 2008.
[20] M. Cardei, M. Thai, Y. Li and W. Wu, "Energy-efficient target coverage in wireless sensor newworks," *IEEE INFOCOM*, 2005.
[21] J. Ai and A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 1, pp. 21-41, 2006.
[22] D. Hall and S. McMullen, *Mathematical Techniques in Multisensor Data Fusion, Artech House*, 1992.
[23] A. Schrijver, *Theory of Linear and Integer Programming, Wiley*, 1998.
[24] IBM CPLEX, *User's Manual for CPLEX, International Business Machines Corporation*, 2009.