Efficient Dispatch of Mobile Sensors in a WSN with Wireless Chargers

You-Chiun Wang and Jiun-Wen Huang

Abstract—Hybrid *wireless sensor networks (WSNs)* contain static and mobile sensors to support various applications. Static sensors monitor environment to find where events occur. *Mobile sensors (MSs)* move to *event locations (ELs)* to do advanced analysis. Since MSs are powered by batteries, how to dispatch them to visit ELs in an energy-efficient manner is critical. Moreover, as the technology of *wireless chargers (WCs)* is mature, it is feasible to use WCs to replenish energy of MSs during their working time. The paper addresses the *multi-round sensor dispatch problem* in a hybrid WSN with WCs, whose goal is to schedule the paths of MSs to visit ELs and WCs in each round, such that their lifetime is maximized. The problem is NP-hard and we propose a *group-based mobile sensor dispatch (G-MSD)* algorithm, which clusters ELs into groups by their positions. G-MSD dispatches MSs to visit each group of ELs to save and balance their moving energy costs. Besides, it schedules MSs to call at WCs to recharge batteries on demand. Simulation results verify that the G-MSD algorithm greatly extends lifetime of MSs, as comparing with other methods.

Index Terms—load balance, mobile sensor, path planning, wireless charger, wireless sensor network.

1 Introduction

A *wireless sensor network (WSN)* is made up of a large number of sensors with capabilities of sensing, computation, and communication. They are deployed in a region of interest to collect environmental information, and report sensing data to a remote sink. Due to their context-aware monitoring ability, WSNs promote the development of IoT (Internet of things) [1], where appliances, devices, or vehicles can organize a network for people to access their data by equipping with sensors. Today, WSNs have been widely used in a variety of applications, from precision agriculture [2] to smart shopping [3], structural detection [4], and pollutant monitoring [5].

As robots and vehicles are mature technologies, it is practical to implement *mobile sensors (MSs)* by installing sensors on these platforms [6]. Introducing mobility to sensors can overcome difficulties incurred by static WSNs. First, since sensors are often randomly deployed, there may be some *coverage holes* without any sensor in the sensing field. When sensors are broken or drained of energy, they leave more holes. Thus, we can eliminate these holes by moving MSs to cover them [7]. Second, it is infeasible to vary the mission of a static WSN without adding new nodes. However, this problem can be simply solved by using MSs. Third, some applications require sophisticated sensors to detect certain events, but it is uneconomic to deploy a lot of such sensors. In this case, we can deploy cheap static sensors to provide basic monitoring of the environment, while use sophisticated MSs to move in the sensing field to support different missions [8].

This paper investigates the problem of dispatching MSs to analyze events occurring in a hybrid WSN. Static sensors form a backbone to find out where events appear and report them to the sink. MSs are dispatched to visit *event locations (ELs)* to perform in-depth analysis. Since events may occur anytime and anywhere, it is inefficient to dispatch an MS right after the occurrence of an event. Thus, we divide time into *rounds* and schedule the movement of MSs in a round-by-round fashion. Since MSs are powered by small batteries and the energy spent on movement dominates their energy consumption [9], we aim at the issues of path efficiency and load balance of MSs. Specifically, our goal is to maximize system lifetime of the hybrid WSN, which is defined by the number of rounds until some ELs cannot be visited by any MS due to exhaustion of energy.

Moreover, wireless charging allows a power source to transmit electromagnetic energy to electrical devices, so there is no need to use cords to connect the power source and devices. There are three popular techniques of wireless charging [10], including inductive coupling, magnetic resonance coupling, and radio frequency radiation. Recently, *wireless chargers (WCs)* are popularly used to recharge the batteries of sensors to prolong their usage time [11], [12]. It motivates us to adopt WCs to help extend system lifetime by adaptively replenishing energy of MSs.

Given a few static WCs, this paper formulates a *multi-round sensor dispatch problem* in a hybrid WSN, which asks how to efficiently schedule the paths of MSs to visit ELs and WCs in each round, such that system lifetime is maximized. The problem is NP-hard, so we propose a *group-based mobile sensor dispatch (G-MSD)* algorithm. Our idea is to divide ELs into groups based on their positions, where ELs close to each other are grouped together. Then, MSs are dispatched to visit each group of ELs with the objectives of reducing moving paths and balancing their energy consumption. Besides, MSs will take time out from their dispatching jobs to recharge batteries by moving to the positions of WCs on their ways to visit ELs. Through simulations, we show that our G-MSD algorithm can significantly extend system lifetime.

We outline this paper as follows: Section 2 discusses related work and Section 3 gives system model. Section 4 formulates our problem. Then, we propose the G-MSD algorithm in Section 5, followed by its performance evaluation in Section 6. Finally, Section 7 concludes this paper.

The authors are with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan. Email: ycwang@cse.nsysu.edu.tw; m043040051@student.nsysu.edu.tw.

2 Related Work

2.1 Mobility Management in Multi-robot Systems

In robotics, it attracts considerable attention to make multiple robots collaborate to carry out certain tasks such as predator avoidance [13] and trajectory tracking [14]. To do so, robots should be trained to learn the mappings between their statuses and actions, and different strategies like Q-learning [15] and neural network [16] are proposed. Although these studies involve in mobility management of robots, they have different objectives with our work.

Task assignment is also critical in multi-robot systems. Each task gives some target sites to be visited by robots, and the goal is to minimize their costs (e.g., reducing moving distances) [17]. This NP-hard problem can be solved by auction-based methods [18], where each robot submits a bid (e.g., its distance to reach a target site) to contest the task. Then, the robot which gives the best bid can take the task. The work [19] uses *combinatorial auctions*, where the auction of a task has multiple items and each robot can bid on the combination of these items. There are some similarities between the task assignment problem and our sensor dispatch problem. However, the auction-based methods select a robot with the best bid to visit a target site. Some robots may take more tasks and fast exhaust energy. On the contrary, our G-MSD algorithm aims to balance energy consumption of MSs to prolong their lifetime.

2.2 Mobility Management in Mobile WSNs

A number of studies use MSs to track objects. For example, [20] moves MSs to improve the monitoring quality of a tracking object, on the premise that MSs will not be disconnected or reduce their coverage due to movement. Tan et al. [21] use a WSN to look for an object, and ask MSs with higher signal-to-noise ratios to move close to the object to provide in-depth detection. In [22], MSs have two roles: leaders and followers. A leader finds out the object in the sensing field, while followers keep connectivity among sensors. Obviously, they cope with different problems with ours.

How to move MSs to adjust WSN topology is also critical, as the distribution of sensors has great impact on its detection ability [23]. The study [24] views MSs as electric charges, so they can exert forces to make themselves be evenly distributed over the sensing field. The work [25] finds the sites to place sensors to fully cover the sensing field, and uses a weighted bipartite graph to decide which MSs should move to these sites. The result is extended to multi-level coverage by [26], and MSs compete to move to target sites in a distributed manner. The work [27] finds coverage holes by a Voronoi diagram, and moves MSs to cover holes. These studies consider saving energy of MSs on movement, but they seek to optimize the result in only one round.

2.3 Sensor Dispatch in Hybrid WSNs

A few sensor dispatch problems are proposed for hybrid WSNs. In [28], static sensors notify nearby MSs of event occurrence and ask them to analyze the event. When multiple MSs get the notification, it chooses the MS that has a shorter moving distance and whose leave causes a smaller coverage hole, to visit the EL. The study [29] dispatches MSs to improve sensing coverage of a hybrid WSN. Static sensors estimate coverage holes and use the sizes of holes to bid for MSs. However, [28], [29] do not consider lifetime of MSs. The work [30] assumes that multiple types of events may occur in the sensing field. The type of each event is single, but an MS can analyze different types of events. The goal is to dispatch MSs to visit ELs such that their types coincide and we can prolong lifetime of MSs. A two-phase solution is proposed, where the first phase finds one-to-one assignments between MSs and ELs, and the next phase creates spanning trees for MSs to visit unassigned ELs. Obviously, this sensor dispatch problem is different from ours.

The work [31] considers a similar sensor dispatch problem without WCs. An *energy-balanced dispatch (EBD)* method is developed, which clusters ELs and assigns MSs to visit each cluster. EBD uses a concept of bound to restrict MSs that can visit each cluster, so as to balance their energy consumption. Our G-MSD algorithm has three differences with EBD. First, EBD clusters ELs only when ELs are more than MSs. In contrast, G-MSD always groups ELs that are close to each other, so as to save moving energy of MSs. Second, G-MSD and EBD have different strategies in choosing MSs, where EBD selects MSs based on merely their energy costs to visit ELs, while G-MSD considers additionally the residual energy of each MS. Thus, we can prevent MSs with less energy from visiting large groups of ELs. Third, G-MSD allows MSs to call at WCs to timely recharge their batteries. These designs distinguish our G-MSD algorithm with the EBD method and give its novelty. Simulation results in Section 6 will also show that G-MSD significantly outperforms EBD in terms of system lifetime.

2.4 Using WCs to Extend WSN Lifetime

How to use WCs to extend lifetime of sensors is widely discussed. Both [12], [32] find where to place WCs in a WSN such that the charging quality is maximized. Since adjacent WCs may pose radio interference with each other, [33] uses a concurrent charging schedule to fully charge sensors in the minimum time. Besides, many studies [11], [34], [35] equip WCs on mobile platforms, and ask them to travel in a WSN to recharge sensors. These studies aim at tour planning of WCs to make them fast visit sensors with less energy. None of them considers using WCs to extend system lifetime in our sensor dispatch problem.

Several studies handle the data gathering problem by using a mobile node to collect data from sensors. The work [36] discusses how to use a mobile sink to travel in a rechargeable WSN to collect data from sensors, where each sensor can harvest energy from its surroundings (e.g., solar and wind). The goal is to maximize the amount of data collected by the mobile sink while keeping network fairness. Obviously, [36] discusses a different problem with our paper, as sensors can generate energy by themselves, not by WCs. Guo et al. [37] use a WC to recharge sensors and also collect their data. A set of sensors are selected as anchors, where other sensors will relay data to these anchors. Then, they decide both sequence and sojourn time for a WC to visit anchors to get data and replenish energy. Unlike [37], our paper addresses how to let MSs move to visit (static) WCs to recharge their batteries, rather than using a mobile WC for energy replenishment. The work [38] deploys both MSs and WCs in a WSN to extend lifetime of sensors, where MSs and WCs move to visit sensors to collect data and recharge batteries, respectively. Nevertheless, WCs replenish energy of only static sensors. MSs will be recharged only when they

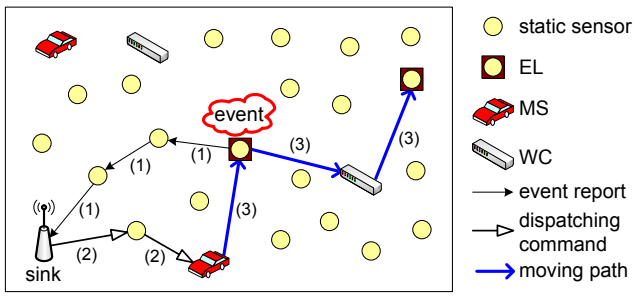


Fig. 1: The sensor dispatch scenario in a hybrid WSN: (1) static sensors notify the sink of ELs, (2) the sink assigns dispatching jobs to MSs, and (3) MSs move to visit ELs and WCs.

return to the sink's location. Thus, [38] has a different energy replenishment model with our sensor dispatch problem.

3 System Model

3.1 Network Model

We consider a hybrid WSN with static and mobile sensors, as shown in Fig. 1. Static sensors form a backbone to connect all nodes in the WSN, so the sink can get data from sensors and send commands to them. When events occur, static sensors notify the sink of their locations. As multiple static sensors may detect the same event, we use the method in [28] to select a delegate to notify the sink. Thus, each event can be modeled by a point in the sensing field.

MSs are randomly scattered over the sensing field. When the sink gets ELs, it can ask MSs to move (with a constant speed) to visit ELs to analyze events. MSs are aware of their locations [39], so they know where to move. If the sensing field is obstacle-free, the shortest distance between an MS and the destination (e.g., EL or WC) is their Euclidean distance. Otherwise, we use the method in [25] to find the shortest path for an MS to reach its destination by detouring obstacles. Besides, MSs directly send their analytic data to the sink via the backbone. Thus, after an MS finishes its dispatching job, it stays at the last EL, instead of going back to the sink's location.

3.2 Energy Model

Each MS has the same battery capacity $e_{\max}$ and energy cost e_{cost} to move a unit distance. Besides, there are fixed WCs in the sensing field, where their locations are known by MSs. Each WC recharges one MS at a time, and it can recharge at most α MSs in a round. An MS has to move to the WC's location to recharge its battery. The charging rate is a constant r [40]. Once the MS is fully charged, it has an amount $e_{\max}$ of energy. As MSs are realized by equipping sensors on mobile platforms, we assume that each MS has separated batteries for its mobile platform (to provide mobility) and sensing device (to support other jobs like event analysis and communication). Since we aim at investigating path efficiency of MSs, both $e_{\max}$ and r are used by mobile platforms.

Let $T_{\max}$ be the maximum amount of time for an MS to do its job (i.e., visiting ELs) in a round. Each MS is allowed to visit a WC for recharging only when it has enough time to complete its assigned work in a round. Suppose that the time that an MS s_j needs to visit all its ELs is T_j^{D}, which also includes the time that s_j analyzes events and reports data. Beside, let T_j^{C} be the sum of the time that s_j moves to visit a WC and the time that s_j waits for service[1]. In theory, s_j can be recharged for an amount of energy:

$$\min\{r \times (\max\{T_{\max} - T_j^{\text{D}} - T_j^{\text{C}}, 0\}), e_{\max} - e_j\}, \quad (1)$$

where e_j is s_j's residual energy. In Eq. 1, s_j has an opportunity to recharge its battery only when $T_j^{\text{D}} + T_j^{\text{C}} < T_{\max}$ (i.e., it has enough time to visit all assigned ELs and also move to meet the WC for service). That is why we use the term $\max\{T_{\max} - T_j^{\text{D}} - T_j^{\text{C}}, 0\}$. Besides, due to the battery capacity, s_j can be recharged with no more than an amount $(e_{\max} - e_j)$ of energy.

One may suggest letting an MS complete its assigned work *after* it visits a WC to replenish energy, thereby relaxing the assumption in Eq. 1. However, this scheme is feasible only when events will last for a long time. Let us consider one special case, where events disappear in the beginning of the next round. If we relax the restriction in Eq. 1, some MSs may spend (much) more time on recharging batteries and have not enough time to visit their assigned ELs in the current round. Thus, we never can dispatch any MS to analyze these events (in the next round), as they have already disappeared. In this case, we fail to solve the sensor dispatch problem, even though all MSs have sufficient energy. That is why we have to consider the restriction of Eq. 1.

4 Multi-round Sensor Dispatch Problem

We are given a set $\hat{S}$ of MSs and a set $\hat{C}$ of WCs. Suppose that a set $\hat{L}$ of ELs are reported by static sensors in each round. The *multi-round sensor dispatch problem* asks how to assign each MS $s_j \in \hat{S}$ a visiting schedule V_j in every round, which contains a sequence of ELs and possibly WCs (i.e., $V_j \subseteq \hat{L} \cup \hat{C}$), such that system lifetime (in rounds) is maximized, under two conditions:

$$\hat{L} \subseteq \bigcup\nolimits_{s_j \in \hat{S}} V_j, \quad (2)$$

$$e_{\text{cost}} \times \Gamma(V_j) \leq e_j + \tilde{E}(s_j, V_j \cap \hat{C}), \quad (3)$$

where $\Gamma(V_j)$ is the length of a shortest path for s_j to visit all nodes in V_j, and $\tilde{E}(s_j, V_j \cap \hat{C})$ gives the amount of energy that s_j is replenished by the WC(s) which it visits (i.e., $V_j \cap \hat{C}$). If $V_j \cap \hat{C} = \emptyset$, we have $\tilde{E}(s_j, V_j \cap \hat{C}) = 0$. Here, Eq. 2 indicates that the union of nodes in all visiting schedules should include $\hat{L}$. In other words, each EL in $\hat{L}$ must be visited by an MS. On the other hand, Eq. 3 means that s_j should have enough energy (including the energy $\tilde{E}(s_j, V_j \cap \hat{C})$ replenished from WCs) to complete its visiting schedule V_j. Obviously, when these two conditions are violated, system lifetime terminates, as we cannot schedule MSs to visit all ELs due to lack of energy.

In our previous work [31], we showed that the problem of dispatching MSs in a hybrid WSN *without* WCs is NP-complete. Since the problem in [31] can be viewed as a special case of the sensor dispatch problem by setting $\hat{C} = \emptyset$, it will be NP-hard. Table 1 summarizes our notations.

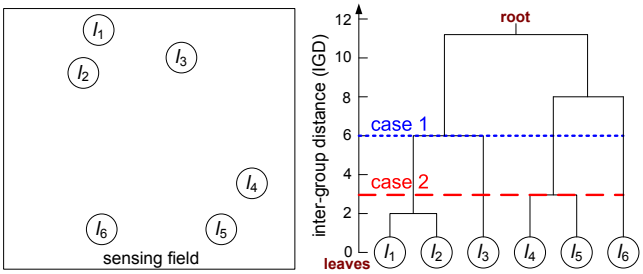
5 The Proposed G-MSD Algorithm

G-MSD contains two schemes to schedule the movement of MSs. In the *task allocation scheme*, we assign each MS some ELs to be visited, so as to reduce their energy consumption and also balance their loads. Then, in the *energy replenishment*

1. When multiple MSs come to visit the same WC, s_j may have to wait others before the WC can serve it.

TABLE 1: Summary of notations.

notations	definitions
$\hat{S}, \hat{C}, \hat{L}, \hat{G}$	sets of MSs, WCs, ELs, and groups
$e_{\max}, e_{\text{cost}}$	battery capacity and moving energy cost
e_j	residual energy of MS s_j
r	charging rate
$p_{i,j}$	priority of MS s_j with respect to group g_i
$\Phi_i^{\text{G}}, \Phi_j^{\text{S}}$	PO lists of group g_i and MS s_j
$T_{\max}$	maximum working time in a round
$T_{j,k}^{\text{A}}, T_{j,k}^{\text{S}}, T_{j,k}^{\text{R}}$	arrival, staying, and residual time in Fig. 5
$T_{j,k}^{\text{EC}}$	recharging time that WC c_k allocates to MS s_j
δ_{IGD}	threshold on the inter-group distance by eAHC
α	number of MSs that a WC can serve in a round
β	a coefficient to adjust limit ε



(a) EL distribution (b) eAHC result

Fig. 2: An example of grouping ELs.

scheme, we let MSs call at WCs to recharge their batteries, such that they will spend less energy to move to visit WCs and the total charging amount of MSs can be maximized. Below, we detail our designs in both schemes.

5.1 Task Allocation Scheme

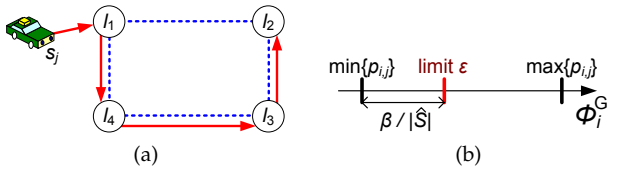
This scheme contains three phases to select an MS to visit each EL. In phase 1, we divide ELs into groups, such that ELs in a group are close to each other. In phase 2, we compute a *priority* for every MS based on its moving cost and residual energy, and create a *preference-order (PO)* list for each group that sorts MSs by their priorities. In phase 3, each group selects an MS from the PO list and contests it with other groups. Below, we detail each phase.

5.1.1 Phase 1: Group ELs

When there are more ELs than MSs, it is efficient to group ELs such that we can find a one-to-one assignment between each MS and each group [31]. In fact, when ELs are fewer than MSs, we can also group those ELs close to each other and ask one MS to visit the group. Thus, we need not dispatch multiple MSs to visit ELs in the group, thereby saving their energy.

To group ELs, we use the *enhanced agglomerative hierarchical clustering (eAHC)* strategy [41]. Each EL $l_i \in \hat{L}$ is initially viewed as a single group g_i. Then, we iteratively merge two groups g_i and g_j such that they have the shortest *inter-group distance (IGD)*, which is the distance between the two farthest ELs l_a and l_b, where $l_a \in g_i$ and $l_b \in g_j$. This iteration is repeated until the minimum IGD exceeds a threshold δ_{IGD}. We give an example in Fig. 2, where $|\hat{L}| = 6$. Fig. 2(a) shows the distribution of ELs in the sensing field, while Fig. 2(b) presents a tree structure that describes the relationship of different groups formed by eAHC.

The grouping result of eAHC depends on δ_{IGD}. In G-MSD, there are two cases to be discussed:



(a) (b)

Fig. 3: Two designs in phase 2: (a) finding the shortest path and (b) computing the limit ε.

- Case of $|\hat{S}| < |\hat{L}|$: As mentioned earlier, we should cluster ELs of $\hat{L}$ into the same number of groups with MSs. Let us consider an example in Fig. 2(b) with $|\hat{S}| = 3$. From the leaves of the eACH's tree, we gradually move towards the root, until we divide $\hat{L}$ into three groups. Thus, we get $\delta_{\text{IGD}} = 6$ and the grouping result is $g_1 = \{l_1, l_2, l_3\}$, $g_2 = \{l_4, l_5\}$, and $g_3 = \{l_6\}$.
- Case of $|\hat{S}| \geq |\hat{L}|$: In this case, we set the IGD threshold by

$$\delta_{\text{IGD}} = \frac{\rho}{|\hat{S}_{\text{FQ}}|} \sum\nolimits_{s_j \in \hat{S}_{\text{FQ}}} \frac{e_j}{e_{\text{cost}}}, \tag{4}$$

where $\hat{S}_{\text{FQ}}$ is a subset of $\hat{S}$ which contains the first quarter of MSs with the least amount of energy, and $0 < \rho < 0.5$ is a coefficient. We give an example in Fig. 2(b), where there are six MSs with energy of $e_1 = 30$, $e_2 = 40$, $e_3 = 45$, $e_4 = 58$, $e_5 = 75$, and $e_6 = 87$. Thus, we have $\hat{S}_{\text{FQ}} = \{s_1\}$. Let us set $e_{\text{cost}} = 3$ and $\rho = 0.3$. Then, the IGD threshold is $\delta_{\text{IGD}} = 0.3 \times \frac{30}{3} = 3$. Thus, the grouping result will be $g_1 = \{l_1, l_2\}$, $g_2 = \{l_3\}$, $g_3 = \{l_4, l_5\}$, and $g_4 = \{l_6\}$.

We discuss the idea behind Eq. 4. When MSs have less energy, we should form smaller groups of ELs so that each MS will not consume too much energy on visiting the ELs in a group. Thus, we estimate the maximum movable distance of each MS by e_j/e_{cost}, and take their average of MSs in $\hat{S}_{\text{FQ}}$ (i.e., the first 25% of MSs with less energy) as a reference. Besides, we use a coefficient ρ to limit the size of groups in Eq. 4, so an MS will not spend all its energy to visit one group of ELs.

5.1.2 Phase 2: Create PO lists

After phase 1, we obtain a set $\hat{G}$ of groups, where any two groups will be disjointed. For each group $g_i \in \hat{G}$, we compute a priority of every MS $s_j \in \hat{S}$ with respect to g_i by

$$p_{i,j} = \xi(g_i, s_j)/e_j, \tag{5}$$

where $\xi(g_i, s_j)$ is the amount of energy that s_j spends to visit all ELs in g_i. Here, a smaller $p_{i,j}$ value means a higher priority. From Eq. 5, when s_j can spend less energy to do the job in g_i or it has more energy, s_j will be given a higher priority (i.e., s_j is more suitable for g_i). Then, we create a PO list Φ_i^{G} for g_i, which sorts MSs by their priorities (from high to low).

To get $\xi(g_i, s_j)$ in Eq. 5, we find the shortest path for s_j to visit all ELs in g_i. However, this method spends much computational time, as we have to use a solution to the NP-complete *traveling salesman problem (TSP)* to find the path for every pair of MSs and groups. To save the computational time, we find the minimum Hamiltonian cycle to visit all ELs in g_i. Then, we find the closest EL, say, l_k in g_i to s_j, and remove the longest edge linking to l_k from the cycle. Thus, the shortest path includes edge (s_j, l_k) and the Hamiltonian path. In this way, we can greatly reduce the computational overhead, as

we find the Hamiltonian cycle only once, instead of repeating the TSP solution $|\hat{S}|$ times for each group. In fact, there have been many heuristics and approximation algorithms proposed to find the Hamiltonian cycle [42]. We give an example in Fig. 3(a), where g_i has four ELs. The minimum Hamiltonian cycle in g_i is $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \rightarrow l_1$. Since l_1 is closest to s_j, we remove edge (l_1, l_2), and the shortest path is $s_j \rightarrow l_1 \rightarrow l_4 \rightarrow l_3 \rightarrow l_2$. Then, $\xi(g_i, s_j)$ can be calculated by taking the product of path length and e_{cost}.

From the PO list Φ_i^{G}, we select candidate MSs for each group g_i by a *limit*:

$$\varepsilon = \frac{1}{|\hat{G}|} \times \sum_{i=1}^{|\hat{G}|} \min_{\Phi_i^{\text{G}}}\{p_{i,j}\} + \frac{\beta}{|\hat{G}| \times |\hat{S}|} \times \sum_{i=1}^{|\hat{G}|} (\max_{\Phi_i^{\text{G}}}\{p_{i,j}\} - \min_{\Phi_i^{\text{G}}}\{p_{i,j}\}), \quad (6)$$

where $\beta < |\hat{S}|$ is a coefficient to adjust the value of limit, and we will discuss its effect in Section 6.6. Then, we mark an MS s_j in Φ_i^{G} as a candidate if $p_{i,j} \leq \varepsilon$. Besides, we use a variable b_i to record the number of candidate MSs in Φ_i^{G}, where g_i will use b_i as the *bid* to contest an MS in Section 5.1.3. Fig. 3(b) shows our calculation of limit ε in Eq. 6, where we take the minimum $p_{i,j}$ value in each PO list as its basis, and add a fixed portion $\beta/|\hat{S}|$ of the difference between the maximum and minimum $p_{i,j}$ values in each PO list to ε. By using the limit ε, we can avoid selecting an MS far away from group g_i to move to visit its ELs, and balance energy consumption of MSs accordingly.

5.1.3 Phase 3: Pair Groups with MSs

In this phase, we pair each group with an MS by the following steps:

1) We compute an energy threshold

$$\delta_E = \mu_E - \sqrt{\frac{1}{|\hat{S}|} \sum\nolimits_{s_j \in \hat{S}} (e_j - \mu_E)^2}, \quad (7)$$

where $\mu_E = (\sum_{s_j \in \hat{S}} e_j)/|\hat{S}|$ is average residual energy of MSs. Here, δ_E is the difference between the average energy and the standard deviation of energy of MSs. Then, we use a queue $\tilde{Q}_{\text{G}}$ to store all groups in $\hat{G}$.

2) Dequeue a group g_i from $\tilde{Q}_{\text{G}}$, and select the first candidate MS, say, s_j from its PO list. If s_j is unpaired, we pair s_j with g_i (i.e., s_j should visit ELs in g_i). Otherwise, s_j must be paired with another group, say, g_k. In this case, we use five rules in sequence to let them contest s_j:

   R1. When $e_j < \delta_E$, we compare the energy costs for s_j to visit ELs in g_i and g_k. Specifically, if $\xi(g_i, s_j) < \xi(g_k, s_j)$, we pair s_j with g_i. Otherwise, s_j is still paired with g_k.

   R2. If $b_i < b_k$, we pair s_j with g_i.

   R3. Suppose that $p_{i,x}$ and $p_{k,y}$ are the lowest priorities in the PO lists of g_i and g_k, respectively. If $b_i = b_k$ and $p_{i,x} > p_{k,y}$, we pair s_j with g_i.

   R4. If $b_i = b_k$, $p_{i,x} = p_{k,y}$, and $\xi(g_i, s_j) < \xi(g_k, s_j)$, we pair s_j with g_i.

   R5. Otherwise, s_j should be still paired with g_k.

   If s_j is paired with g_i, we add g_k to $\tilde{Q}_{\text{G}}$ (as it becomes unpaired), update its PO list Φ_k^{G} and bid b_k by the rule in step 3, and go to step 4. Otherwise, we go to step 3 to deal with g_i.

3) Since s_j is not paired with g_i, we remove s_j from Φ_i^{G} and decrease its bid b_i by one. If $b_i = 0$ (i.e., there is no candidate MS), we mark the first MS in Φ_i^{G} as a candidate and set $b_i = 1$.

4) Repeat step 2 until $\tilde{Q}_{\text{G}}$ becomes empty.

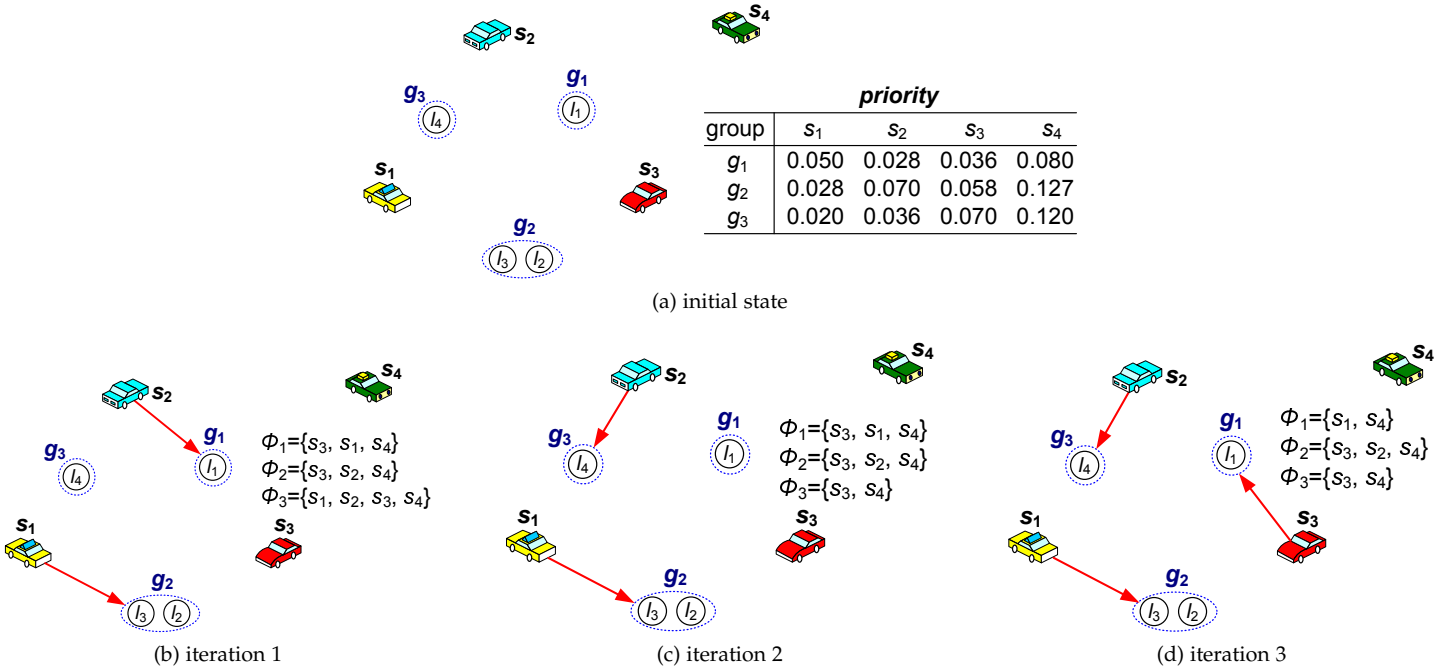
Thanks to the grouping operation in Section 5.1.1, we have $|\hat{G}| \leq |\hat{S}|$. Thus, we ensure that each group in $\hat{G}$ can always find an MS in $\hat{S}$ to visit its ELs. After assigning an MS s_j to a group g_i, s_j can visit all ELs in g_i following the shortest path calculated in Section 5.1.2.

Fig. 4 gives an example, where $\hat{S} = \{s_1, s_2, s_3, s_4\}$ and $\hat{G} = \{g_1, g_2, g_3\}$. Based on the priorities in Fig. 4(a), the PO lists of g_1, g_2, and g_3 are $\Phi_1^{\text{G}} = \{s_2, s_3, s_1, s_4\}$, $\Phi_2^{\text{G}} = \{s_1, s_3, s_2, s_4\}$, and $\Phi_3^{\text{G}} = \{s_1, s_2, s_3, s_4\}$, respectively. By setting $\beta = 10$ in Eq. 6, we can get the limit $\varepsilon = 0.2345$. Thus, the candidates of each group contain all MSs in $\hat{S}$. In the beginning, we use a queue $\tilde{Q}_{\text{G}}$ to include all groups in $\hat{G}$. Then, we dequeue g_1 from $\tilde{Q}_{\text{G}}$ and pair it with the first candidate in Φ_1^{G} (i.e., s_2). Similarly, we also pair g_2 with s_1, as shown in Fig. 4(b). After dequeuing g_3 from $\tilde{Q}_{\text{G}}$, we find that its first candidate s_1 has been paired with g_2. In this case, g_3 contests s_1 with g_2. Since $b_2 = 3$ and $b_3 = 4$, g_2 wins the contest by rule R2. Thus, g_3 removes s_1 from its PO list Φ_3^{G} and selects the next candidate s_2. However, s_2 has been paired with g_1, so g_3 has to contest s_2 with g_1. Since the lowest priorities in Φ_1^{G} and Φ_3^{G} are 0.08 and 0.12, respectively, g_3 wins the contest by rule R3 and we thus pair s_2 with g_3, as shown in Fig. 4(c). Then, g_1 is added to $\tilde{Q}_{\text{G}}$ again. Finally, we pair g_1 with s_3 from its PO list Φ_1^{G}, as shown in Fig. 4(d).

We discuss our design in the five rules. In rule R1, when the amount of residual energy of the contested MS s_j is below the energy threshold δ_E, it means that s_j may not move in a long distance. Thus, we should choose the group with a lower energy cost $\xi(g_i, s_j)$ for s_j to do the job. Rules R2–R5 are designed for the case of $e_j \geq \delta_E$ (i.e., s_j has more energy). When two groups g_i and g_k contest s_j, we first compare their numbers of candidate MSs by rule R2. If a group has more candidates, we make it give up s_j, as there is a good possibility for the group to choose another MS. However, when g_i and g_k have the same number of candidates, we compare their lowest priorities by rule R3. If a group has a lower priority (i.e., larger $p_{i,j}$ value), it means that once the group gives up s_j, the group may choose another MS far away from it (referring to Eq. 5). In this case, we would pose unbalanced loads on MSs. That is why we let the group with a lower priority win the contest. When g_i and g_k have the same (lowest) priority, we choose the group with a lower energy cost to be the winner by rules R4 and R5, so as to reduce energy consumption of s_j.

5.1.4 Discussion

The idea of our task allocation scheme is to group ELs and allow these groups to contest MSs with the objectives of minimizing energy consumption and balancing their loads. There are three special designs in the scheme. First, we group ELs that are close to each other by the eAHC strategy, even if there are fewer ELs than MSs. Moreover, we use a threshold δ_{IGD} to decide the diameter of each group by Eq. 4, which considers residual energy of MSs. Thus, we can dispatch an MS to visit nearby ELs, instead of dispatching multiple MSs to visit them (which wastes energy). Second, when selecting MSs



	priority			
group	s_1	s_2	s_3	s_4
g_1	0.050	0.028	0.036	0.080
g_2	0.028	0.070	0.058	0.127
g_3	0.020	0.036	0.070	0.120

(a) initial state

(b) iteration 1 (c) iteration 2 (d) iteration 3

Fig. 4: An example of pairing groups with MSs.

to visit groups, we adopt both the energy cost $\xi(g_i, s_j)$ and residual energy e_j of each MS as the metric by Eq. 5. Thus, we can avoid dispatching an MS with less energy to visit a group that requires a longer moving distance or contains more ELs. So, the MS can conserve energy and we also let it visit a WC to replenish energy by the scheme in Section 5.2. Third, our task allocation scheme uses the energy threshold δ_E to let an MS with less energy choose the task with a lower energy cost to save its energy. Then, we allow two groups contesting an MS based on multiple factors, including the number of candidates, the minimum priority, and the energy cost. Theorem 1 analyzes the time complexity of this scheme.

Theorem 1. *Given m ELs and n MSs, the time complexity of the task allocation scheme is $O(m^2 + (m-n+1)^3 + n^2 \lg n)$ in the worst case, where $m > n$.*

Proof. Our task allocation scheme contains three phases. In phase 1, we use eAHC to group ELs. The worst case occurs when $|\hat{L}| > |\hat{S}|$ (i.e., $m > n$). By using the implementation in [43], it takes $O(m^2)$ to get the result of eAHC. To find the IGD threshold in Eq. 4, we sort all MSs in $\hat{S}$ and compute the average energy of MSs in $\hat{S}_{\text{FQ}}$, so it spends time of $O(n \lg n) + O(n/4)$. Thus, phase 1 takes time of $O(m^2) + O(n \lg n) + O(n/4) = O(m^2)$.

In phase 2, we compute the priority of each MS with respect to each group. Since $m > n$, there will be n groups found in phase 1. The worst case occurs when one group has $(m-n+1)$ ELs while each of other $(n-1)$ groups has an EL. It takes time of $(n-1) \times O(n) = O(n(n-1))$ to compute the priorities of MSs for these $(n-1)$ groups by Eq. 5. For the group with $(m-n+1)$ ELs, we use the Christofides algorithm [42] to find the minimum Hamiltonian cycle to visit all ELs, which takes $O((m-n+1)^3)$ time. We also find the closest EL in the group for each MS, which spends $O(n(m-n+1))$ time. Thus, it takes time of $O((m-n+1)^3)+O(n(m-n+1)) = O((m-n+1)^3)$ to compute the priorities of MSs for the group with $(m-n+1)$ ELs. To get the PO list for each group, we sort all MSs by their priorities. This operation takes time of $n \times O(n \lg n) = O(n^2 \lg n)$. Then, it spends $O(n)$ time to find the limit ε by Eq. 6. To get candidates in each group, we search MSs in its PO list in sequence. This operation takes time of $n \times O(n) = O(n^2)$. Thus, phase 2 spends time of $O(n(n-1)) + O((m-n+1)^3) + O(n^2 \lg n) + O(n) + O(n^2) = O((m-n+1)^3) + O(n^2 \lg n)$.

In phase 3, we spend $O(2n)$ time to compute the threshold δ_E (i.e., $O(n)$ time to find the average μ_E and $O(n)$ time to find the standard deviation). Then, we use five rules to pair each group with an MS. The worst case occurs when we try all combinations of groups and MSs, which spends $O(n^2)$ time. Thus, phase 3 takes time of $O(2n) + O(n^2) = O(n^2)$.

To sum up, the task allocation scheme will spend time of $O(m^2) + O((m-n+1)^3) + O(n^2 \lg n) + O(n^2) = O(m^2 + (m-n+1)^3 + n^2 \lg n)$ in the worst case, which verifies the theorem. □

5.2 Energy Replenishment Scheme

After assigning each MS s_j to visit ELs, if s_j still has time left in the round, we can let it call at a WC to replenish energy. To improve the recharging efficiency, we consider three objectives: 1) maximize the total amount of replenished energy, 2) minimize the amount of energy spent by each MS to visit a WC, and 3) maximize the number of recharged MSs. With the objectives, our energy replenishment scheme contains three parts to select MSs to visit WCs. In part 1, we create a PO list Φ_j^{S} for each MS s_j to store candidate WCs. In part 2, we pick one WC from Φ_j^{S} to recharge s_j. Besides, we use a list for each WC to record MSs that it needs to serve. In phase 3, we find the recharging sequence of MSs based on their arrival time at a WC. Below, we detail each part.

5.2.1 Find Candidate WCs

For each MS s_j, we first check if it needs to be recharged or not. When $e_j/e_{\max} \geq \delta_{\text{full}}$, it means that s_j's battery is almost full, where δ_{full} is an upper-bound threshold on the battery's

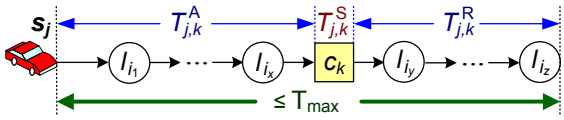


Fig. 5: The amount of time $T^{\mathrm{A}}_{j,k}$, $T^{\mathrm{S}}_{j,k}$, and $T^{\mathrm{R}}_{j,k}$ for an MS s_j.

capacity[2]. In this case, s_j need not be recharged. Otherwise, we find candidate WCs for s_j to recharge its battery. Specifically, we consider two types of MSs below.

Type 1: s_j is assigned with a group g_i of ELs, whose shortest path $H_{i,j}$ is $s_j \rightarrow l_{i_1} \cdots \rightarrow l_{i_z}$. For each WC $c_k \in \hat{C}$, we use the *insertion TSP heuristic* [42] to add c_k to path $H_{i,j}$. Let the new shortest path $H^{\mathrm{A}}_{i,j}$ be $s_j \rightarrow l_{i_1} \rightarrow \cdots \rightarrow l_{i_x} \rightarrow c_k \rightarrow l_{i_y} \rightarrow \cdots \rightarrow l_{i_z}$. We then compute the following time for s_j, as shown in Fig. 5: 1) $T^{\mathrm{A}}_{j,k}$: the amount of time elapsed before s_j arrives at c_k's position (i.e., the amount of time for s_j moves along path $s_j \rightarrow l_{i_1} \rightarrow \cdots \rightarrow l_{i_x} \rightarrow c_k$), 2) $T^{\mathrm{S}}_{j,k}$: the amount of time that s_j stays at c_k's position for recharging, and 3) $T^{\mathrm{R}}_{j,k}$: the amount of time since s_j leaves c_k's position until it visits the last EL (i.e., the amount of time for s_j moves along path $c_k \rightarrow l_{i_y} \rightarrow \cdots \rightarrow l_{i_z}$). Since the maximum working time in a round is $T_{\max}$, we derive that $T^{\mathrm{S}}_{j,k} = T_{\max} - T^{\mathrm{A}}_{j,k} - T^{\mathrm{R}}_{j,k}$. Then, we use two conditions to check if c_k is a candidate of s_j:

$$e_j \geq e_{\mathrm{cost}} \times \mathrm{length}(H^{\mathrm{A}}_{i,j}), \tag{8}$$

$$rT^{\mathrm{S}}_{j,k} \geq e_{\mathrm{cost}} \times \mathrm{length}(l_{i_x} \rightarrow c_k \rightarrow l_{i_y}) + \sigma, \tag{9}$$

where $\mathrm{length}(\cdot)$ denotes the length of a path. Here, Eq. 8 indicates that s_j should have enough energy to move along the amended path $H^{\mathrm{A}}_{i,j}$ that includes c_k. If Eq. 8 is violated, it is infeasible for s_j to visit c_k (due to lack of energy). On the other hand, the left term of Eq. 9, $rT^{\mathrm{S}}_{j,k}$, gives the amount of replenished energy of s_j from c_k, while its right term, $e_{\mathrm{cost}} \times \mathrm{length}(l_{i_x} \rightarrow c_k \rightarrow l_{i_y})$, gives the extra energy cost for s_j to visit c_k. Thus, Eq. 9 indicates that the *net* amount of energy replenishment of s_j from c_k should be at least σ. Otherwise, it is uneconomic for s_j to move to visit c_k for recharging its battery. In our simulations, we set σ to one unit of energy. When both conditions in Eqs. 8 and 9 are satisfied, c_k will be added to s_j's PO list Φ^{S}_j.

Type 2: s_j has no dispatching job in this round. Thus, $T^{\mathrm{A}}_{j,k}$ will be the time for s_j moving from its current position to c_k's position, and $T^{\mathrm{R}}_{j,k} = 0$ as s_j need not visit any EL. Similarly, we also use two conditions to check if c_k is a candidate of s_j:

$$e_j \geq e_{\mathrm{cost}} \times \mathrm{length}(s_j \rightarrow c_k), \tag{10}$$

$$rT^{\mathrm{S}}_{j,k} \geq e_{\mathrm{cost}} \times \mathrm{length}(s_j \rightarrow c_k) + \sigma, \tag{11}$$

If both conditions in Eqs. 10 and 11 are satisfied, c_k is added to s_j's PO list Φ^{S}_j.

After checking all WCs, we sort the candidates in Φ^{S}_j by their $T^{\mathrm{A}}_{j,k}$ values (from small to large). If the PO list of an MS is empty, it will not participate in the energy replenishment scheme.

5.2.2 Assign WCs to MSs

For each WC c_k, we use a serving list ϕ_k to record the set of MSs to be recharged. Let queue $\tilde{Q}_{\mathrm{S}}$ store MSs selected for c_k in Section 5.2.1. We then create ϕ_k by four steps:

2. According to [44], when a battery is being recharged, if the battery has 100% energy (with a tolerance band of 5%), a battery-full event will be triggered to stop the recharging procedure. Thus, we suggest setting $\delta_{\mathrm{full}} = 0.95$.

1) Dequeue an MS s_j from $\tilde{Q}_{\mathrm{S}}$ and pick the first candidate[3], say, c_k from its PO list Φ^{S}_j. We then remove c_k from Φ^{S}_j. Besides, we add s_j to ϕ_k based on its $T^{\mathrm{A}}_{j,k}$ value (from small to large). When $\tilde{Q}_{\mathrm{S}} = \emptyset$, we terminate this method, as we have checked all MSs.
2) Allocate an amount of recharging time for an MS s_j in ϕ_k by

$$T^{\mathrm{EC}}_{j,k} = \frac{e_{\max} - e_{j,k}}{\sum_{s_i \in \phi_k}(e_{\max} - e_{i,k})} \times T^{\mathrm{C}}_{\mathrm{avg}},$$

$$T^{\mathrm{C}}_{\mathrm{avg}} = \frac{1}{|\phi_k|} \sum_{s_i \in \phi_k} T^{\mathrm{S}}_{i,k}. \tag{12}$$

Here, $e_{j,k}$ is the amount of s_j's residual energy when it arrives at c_k's position.
3) With $T^{\mathrm{EC}}_{j,k}$, we can estimate the amount of s_j's replenished energy from c_k by $e^{\mathrm{R}}_{j,k} = r \times T^{\mathrm{EC}}_{j,k}$. Besides, we also compute its total working time T_j (including the time to visit ELs and also the time to wait and recharge at c_k) by the method in Section 5.2.3.
4) We use three conditions to check whether c_k is capable of serving all MSs in its list ϕ_k:

$$|\phi_k| \leq \alpha, \tag{13}$$

$$T_j \leq T_{\max}, \ \forall s_j \in \phi_k, \tag{14}$$

$$e^{\mathrm{R}}_{j,k} - \zeta(s_j, c_k) \geq \sigma, \ \forall s_j \in \phi_k, \tag{15}$$

where $\zeta(s_j, c_k)$ is the energy cost for s_j to visit c_k. In particular, $\zeta(s_j, c_k) = e_{\mathrm{cost}} \times \mathrm{length}(l_{i_x} \rightarrow c_k \rightarrow l_{i_y})$ if s_j belongs to type 1, and $\zeta(s_j, c_k) = e_{\mathrm{cost}} \times \mathrm{length}(s_j \rightarrow c_k)$ otherwise. Eq. 13 indicates that c_k can serve at most α MSs, and Eq. 14 means that total working time of s_j cannot exceed $T_{\max}$ in a round. Similar to Eqs. 9 and 11, Eq. 15 indicates that the net amount of energy replenishment of s_j from c_k should be at least σ. If any condition is violated, we remove the last MS s_l (i.e., with the largest $T^{\mathrm{A}}_{l,k}$ time) from ϕ_k, add s_l to $\tilde{Q}_{\mathrm{S}}$, and go back to step 2 to recompute $T^{\mathrm{EC}}_{j,k}$. Otherwise, we return to step 1 to pick the next MS in $\tilde{Q}_{\mathrm{S}}$.

5.2.3 Decide the Recharging Sequence

Given the serving list ϕ_k of a WC c_k, we compute the recharging sequence (denoted by ϕ^{S}_k) of all MSs in ϕ_k by the following three steps:

1) Dequeue the first MS, say, s_j from ϕ_k. We then compute the waiting time before c_k finishes recharging s_j's battery by $T_{\mathrm{wait}} = T^{\mathrm{A}}_{j,k} + T^{\mathrm{EC}}_{j,k}$. Here, the waiting time is the sum of s_j's arrival time $T^{\mathrm{A}}_{j,k}$ and its recharging time $T^{\mathrm{EC}}_{j,k}$. We can also compute its total working time by

$$T_j = T_{\mathrm{wait}} + T^{\mathrm{R}}_{j,k}. \tag{16}$$

Then, we add s_j to ϕ^{S}_k.
2) Create a list ϕ_{temp}, and move MSs that meet the condition of $T^{\mathrm{A}}_{j,k} \leq T_{\mathrm{wait}}$ from ϕ_k to ϕ_{temp}. Then, we pick MS s_j from ϕ_{temp} with the largest value of $(T^{\mathrm{EC}}_{j,k} + T^{\mathrm{R}}_{j,k})$, add s_j to ϕ^{S}_k, update

$$T_{\mathrm{wait}} = T_{\mathrm{wait}} + T^{\mathrm{EC}}_{j,k}, \tag{17}$$

3. In case that s_j has no candidate (i.e., $\Phi^{\mathrm{S}}_j = \emptyset$), we remove s_j from $\tilde{Q}_{\mathrm{S}}$ and select the next MS.

and set T_j by Eq. 16. Note that since T_{wait} is changed by Eq. 17, we need to update both ϕ_k and ϕ_{temp} by the above rule. This step is repeated until ϕ_{temp} becomes empty.

3) Repeat both steps 1 and 2, until ϕ_k is empty.

Since all MSs in ϕ_k are sorted by their $T^{\text{A}}_{j,k}$ values (from small to large), the first MS s_j in ϕ_k must be also the first MS that arrives at c_k's position. Besides, as c_k can serve only one MS at a time, all other MSs have to wait until c_k finishes recharging s_j (for $T^{\text{EC}}_{j,k}$ time). That is why we compute the waiting time T_{wait} by Eq. 16. In step 2, when an MS satisfies the condition of $T^{\text{A}}_{j,k} \leq T_{\text{wait}}$, it means that the MS has to wait when it arrives at c_k's position (as c_k is still recharging the battery of another MS). Thus, we use a list ϕ_{temp} to store such MSs. Once c_k finishes its recharging job, we pick the MS that will spend the most time to finish its dispatching job (including the recharging time $T^{\text{EC}}_{j,k}$ and the time $T^{\text{R}}_{j,k}$ to visit residual ELs) to be served by c_k (otherwise, the MS may not finish its job in a round). After we decide the next MS to be recharged, the waiting time will be updated by adding the recharging time $T^{\text{EC}}_{j,k}$ of the selected MS, as shown in Eq. 17. Thus, we need to update ϕ_{temp} by including extra MSs whose $T^{\text{A}}_{j,k}$ time do not exceed the new waiting time T_{wait}. In this way, we can decide the recharging sequence of MSs in ϕ_k.

We consider an example with $\phi_k = \{s_1, s_2, s_3\}$, where $(T^{\text{A}}_{j,k}, T^{\text{EC}}_{j,k}, T^{\text{R}}_{j,k})$ of s_1, s_2, and s_3 are (40, 20, 12), (50, 16, 20), and (55, 18, 24), respectively. By step 1, we dequeue s_1 from ϕ_k and add it to ϕ^{S}_k. We compute the waiting time by $T_{\text{wait}} = 40 + 20 = 60$ and also s_1's working time by $T_1 = 60 + 12 = 72$. Since s_2 and s_3 satisfy the condition of $T^{\text{A}}_{j,k} \leq 60$, they are moved from ϕ_k to ϕ_{temp} by step 2. Because $T^{\text{EC}}_{2,k} + T^{\text{R}}_{2,k} = 16 + 20 = 36$ and $T^{\text{EC}}_{3,k} + T^{\text{R}}_{3,k} = 18 + 24 = 42$, we add s_3 to ϕ^{S}_k, update the waiting time by $T_{\text{wait}} = 60 + 18 = 78$, and compute its working time by $T_3 = 78 + 24 = 102$. Then, we add s_2 to ϕ^{S}_k and compute its working time by $T_2 = (78 + 16) + 20 = 114$. The recharging sequence of ϕ_k will be $s_1 \Rightarrow s_3 \Rightarrow s_2$.

5.2.4 Discussion

In this scheme, we analyze the arrival time $T^{\text{A}}_{j,k}$, staying time $T^{\text{S}}_{j,k}$, and residual time $T^{\text{R}}_{j,k}$ of each MS s_j to visit a WC c_k by Fig. 5. Then, we use Eqs. 8–11 to eliminate the WCs that s_j cannot reach or has to spend more energy than it can be recharged from its candidate list. Afterwards, we let each MS select the first WC from its PO list Φ^{S}_j to meet objective 2 in Section 5.2. Besides, each WC also keeps a list ϕ_k to record the set of MSs to be served. Since the total recharging time is limited to $T^{\text{C}}_{\text{avg}}$, we should allocate it to as more MSs as possible to satisfy both objectives 1 and 3. To do so, we use Eq. 12 to allocate the recharging time for each MS in ϕ_k, where an MS with less energy can be recharged for longer time. Moreover, we also use the conditions in Eqs. 13–15 to avoid adding too many MSs to the serving list of a WC. To check the condition in Eq. 14, we use the method in Section 5.2.3 to decide the recharging sequence of MSs. Thus, MSs can move to meet WCs on their ways to visit ELs for energy replenishment, thereby extending system lifetime. Theorem 2 analyzes the time complexity of our energy replenish scheme.

Theorem 2. *Given m ELs, n MSs, and h WCs, the energy replenishment scheme has time complexity of $O(h(m + n(\lg h + \alpha^2)))$ in the worst case.*

TABLE 2: Comparison on the time complexity of different methods.

method	time complexity
EBD (K-means)	$O(mn\varphi) + O(mn \lg n)$
EBD (balanced)	$O(mn\varphi + m^2n^2) + O(mn \lg n)$
EBD (eAHC)	$O(m^2) + O(mn \lg n)$
G-MSD (task allocation)	$O(m^2 + (m - n + 1)^3 + n^2 \lg n)$
G-MSD (energy replenishment)	$O(h(m + n(\lg h + \alpha^2)))$

Proof. In part 1, we use $O(n)$ time to check the status of each MS's battery. Then, we find parameters $T^{\text{A}}_{j,k}$, $T^{\text{S}}_{j,k}$, and $T^{\text{R}}_{j,k}$ for each (MS, WC) pair. The worst case occurs when all MSs belong to type 1. As mentioned in Theorem 1, we consider the case that a group has $(m - n + 1)$ ELs while each of other $(n - 1)$ groups has one EL. For each WC, it takes time of $O(m - n + 1) + O(n - 1) = O(m)$ to find the parameters for all MSs. As there are h WCs, the operation takes $O(mh)$ time. Then, we spend $O(nh \lg h)$ time to sort the PO list of each MS. Thus, the time complexity of part 1 is $O(n) + O(mh) + O(nh \lg h) = O(mh) + O(nh \lg h)$.

In part 2, we compute $T^{\text{EC}}_{j,k}$ for each MS in the serving list of a WC by Eq. 12. Since a WC can serve at most α MSs, step 3 takes $O(2\alpha)$ time. In step 5, we use three conditions to check if a WC can serve all MSs in its list. Here, Eq. 13 takes only $O(1)$ time, and Eq. 15 spends $O(\alpha)$ time. For Eq. 14, we use the method in Section 5.2.3 (i.e., part 3) to compute the working time of an MS. Thus, part 3 takes $O(\alpha)$ time, as it iteratively checks each MS in the serving list. Since we check every MS by Eq. 14, it spends time of $\alpha \times O(\alpha) = O(\alpha^2)$. So, an iteration of part 2 takes time of $O(2\alpha) + O(1) + O(\alpha^2) + O(\alpha) = O(\alpha^2)$. As we check each (MS, WC) pair, there are at most nh iterations. Thus, part 2 spends time of $nh \times O(\alpha^2) = O(nh\alpha^2)$.

Since the calculation of part 3 is included in part 2 (i.e., Eq. 14), the energy replenishment scheme has time complexity of $O(mh) + O(nh \lg h) + O(nh\alpha^2) = O(h(m + n(\lg h + \alpha^2)))$. □

6 Performance Evaluation

We develop a simulator in Java to evaluate performance of G-MSD. The sensing field is a 450 m × 300 m rectangle, where 400 static sensors are deployed. The communication range of a sensor is 80 m, so all static sensors form a connected network. For MSs, we set their parameters based on the configuration of Khepera IV mobile robots [45]. The moving speed is 1 m/s and the energy cost e_{cost} to move one unit distance is 10.75 J/m. Besides, the maximum battery capacity $e_{\max}$ is set to $3600\,\text{s} \times (3400\,\text{mah}/1000) \times 7.4\,\text{V} = 90576\,\text{J}$. There are also four WCs in the network, each deployed on the center of a quarter of the sensing field. The charging rate r is 5 J/s [40]. We set $\delta_{\text{full}} = 0.95$, so an MS will not be recharged if it has more than 95% energy. Besides, by setting $\alpha = 5$, a WC can serve no more than five MSs in a round. In each round, the maximum working time $T_{\max}$ is set to 800 seconds. A number of ELs will be arbitrarily selected from the positions of all static sensors. When an MS arrives at one EL, it will spend 30 seconds to analyze the event.

We compare our G-MSD algorithm with the EBD method [31], which considers two grouping strategies to cluster ELs. In the K-means strategy, ELs are divided into K groups such that each EL in a group is closest to the group's centroid than the centroid of another group, and K is equal to the number of MSs. The balanced strategy modifies the result of K-means by minimizing the difference between the lengths of shortest

TABLE 3: Comparison on system lifetime (in rounds).

number of ELs	without WCs			with WCs		
(i.e., $\|\hat{L}\|$)	40	70	100	40	70	100
EBD (K-means)	240.6	93.6	82.7	429.3	110.7	95.3
EBD (balanced)	241.8	113.8	101.3	438.6	141.9	121.4
EBD (eAHC)	241.7	112.3	110.1	438.1	139.9	133.8
G-MSD	271.7	115.0	112.5	613.8	142.6	137.1

paths to visit all ELs in any two groups. Besides, we also apply the eAHC strategy to EBD to evaluate its effect. In G-MSD, we set $\beta = 15$. For each experiment, we repeat 100 times of simulations and take their average value. Given m ELs, n MSs, and h WCs, Table 2 lists time complexity of different methods, where φ is the number of iterations to perform K-means for clustering (which usually ranges from tens to hundreds).

6.1 System Lifetime

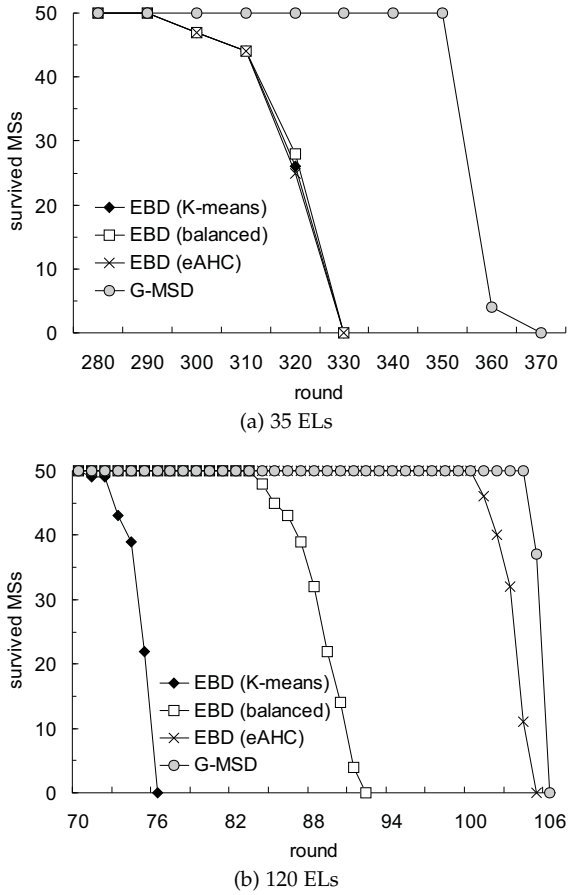
We first measure system lifetime with 50 MSs. The left part of Table 3 gives the experimental result when there are no WCs. For the EBD method, since it does not group ELs when $|\hat{L}| \leq |\hat{S}|$, using different group strategies has less effect on lifetime as $|\hat{L}| = 40$. However, when ELs are more than MSs, the K-means strategy results in shorter lifetime, as comparing with other two strategies. The reason is that the K-means strategy may not form groups of ELs with similar sizes, so some MSs would have to spend more energy to visit ELs in large groups. When $|\hat{L}| = 100$, the eAHC strategy performs better than the K-means and balanced strategies, since it considers the distance between any two adjacent ELs to form groups. Our G-MSD algorithm not only group ELs by eAHC even if $|\hat{L}| \leq |\hat{S}|$ but also considers both energy costs and residual energy of MSs, so it always has the longest lifetime. Specifically, G-MSD can extend 12.5%, 7.9%, and 14.8% of lifetime than EBD when there are 40, 70, and 100 ELs, respectively.

The right part of Table 3 shows system lifetime by using four WCs. Since the EBD method does not consider recharging MSs, we apply our energy replenishment scheme to it. Obviously, we can extend lifetime of MSs by using WCs to recharge them, where the energy replenishment scheme increases 80.0%, 22.7%, and 19.2% of EBD's lifetime with 40, 70, and 100 ELs, respectively. When $|\hat{L}| = 40$, G-MSD can greatly extend lifetime from 271.7 rounds to 613.8 rounds (with an improving ratio of 125.9%). Even when there are 70 and 100 ELs, using WCs can extend averagely 24.0% and 21.9% of G-MSD's lifetime. This experiment shows that both task allocation and energy replenishment schemes in G-MSD can well cooperate, thereby achieving much longer lifetime.

6.2 Survived MSs

We then study survived MSs in each round, where $|\hat{S}| = 50$ and $|\hat{C}| = 0$. Fig. 6(a) gives the result when $|\hat{L}| = 35$. In this case, the grouping strategy has no effect on survived MSs in the EBD method, as it does not group ELs when $|\hat{L}| \leq |\hat{S}|$. In EBD, some MSs use up their energy in the 290th round, and all MSs die after the 330th round. On the other hand, G-MSD can keep all MSs alive until the 350th round, and the longest survived time of MSs is 370 rounds. The reason is that G-MSD considers residual energy when pairing them with groups of ELs. Thus, it will avoid selecting an MS with less energy to visit a large group, thereby extending the MS's lifetime.

Fig. 6(b) shows the result when $|\hat{L}| = 120$. As there are more ELs than MSs, the grouping strategy plays a key role in



(a) 35 ELs

(b) 120 ELs

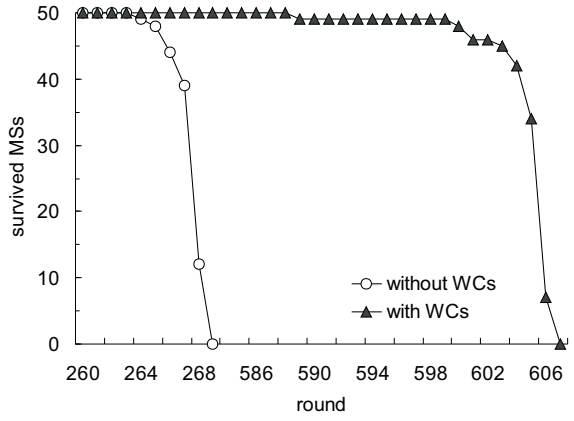
Fig. 6: Comparison on survived MSs by different methods without WCs.

EBD. Specifically, all MSs run out of their energy in the 76th, 92nd, and 106th rounds by EBD when the K-means, balanced, and eAHC strategies are adopted, respectively. Since the G-MSD algorithm uses the eAHC strategy to group ELs and there are more ELs than MSs, G-MSD has similar survived time of MSs with EBD that uses the eAHC strategy. However, the first MS dies in the 100th and 104th rounds in EBD (with eAHC) and G-MSD, respectively, which shows that G-MSD can better utilize energy of MSs.

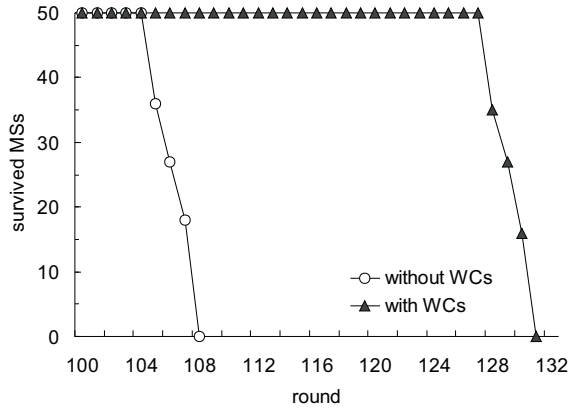
We also measure the effect of WCs on survived MSs in G-MSD, as shown in Fig. 7. When $|\hat{L}| = 40$ (i.e., $|\hat{S}| > |\hat{L}|$), using WCs can greatly extend the survived time of MSs from 269 rounds to 607 rounds (with an improving ratio of 125.7%). Besides, when $|\hat{L}| = 100$ (i.e., $|\hat{S}| < |\hat{L}|$), using WCs can also extend the survived time of MSs from 108 rounds to 131 rounds (with an improving ratio of 21.3%). Note that the number of survived MSs is much larger in the case of $|\hat{S}| > |\hat{L}|$ compared to the case of $|\hat{S}| < |\hat{L}|$. The reason is that each MS is assigned with fewer ELs (or even no ELs) when $|\hat{S}| > |\hat{L}|$. Thus, MSs need not move in longer distances and conserve their energy. Moreover, there is a good possibility for most MSs to call at WCs, as they have more T_j^{S} time (referring to Fig. 5) for recharging. Thus, each MS can survive in a long time when $|\hat{S}| > |\hat{L}|$.

6.3 Energy Consumption

Next, we evaluate energy consumption of MSs without WCs. Fig. 8(a) gives average energy consumed by MSs in a round. Since there are 50 MSs, there is a turning point on the result of each method when $|\hat{L}| = 50$. In particular, as $|\hat{L}|$ grows from 30

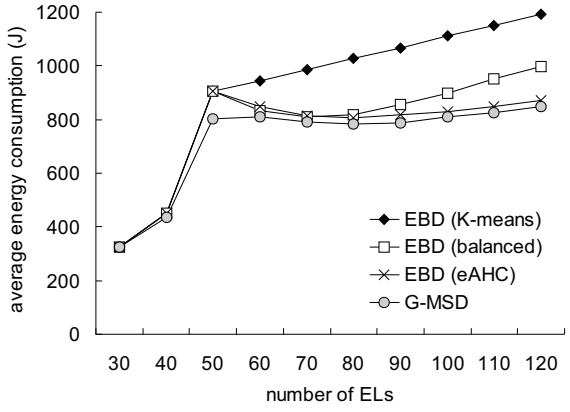


(a) 40 ELs

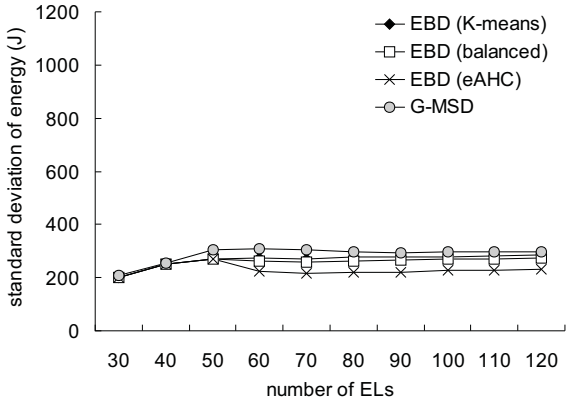


(b) 100 ELs

Fig. 7: Comparison on survived MSs by our G-MSD algorithm without and with WCs.



(a) average



(b) standard deviation

Fig. 8: Comparison on energy consumption of MSs per round.

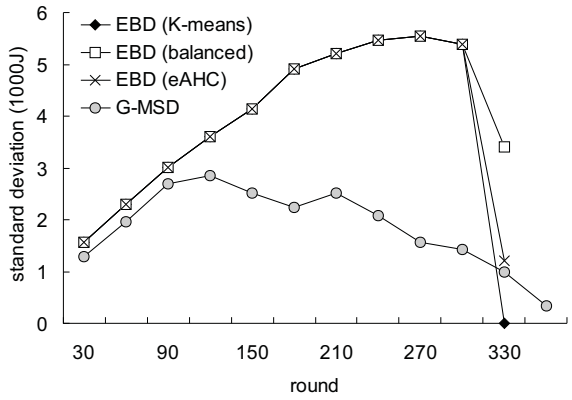
to 50, average energy increases drastically. However, when $|\hat{L}|$ increases from 50 to 120, average energy increases smoothly (since all methods group ELs in the beginning). In the EBD method, the K-means strategy makes MSs consume the most energy, because it would form groups of ELs with unbalanced sizes. Our G-MSD algorithm always lets MSs spend the least energy by dispatching them based on their priorities in Eq. 5, which considers not only the energy cost but also residual energy of each MS. Then, Fig. 8(b) compares the standard deviation of energy consumption by different methods. Since EBD aims to balance energy consumption of MSs while G-MSD considers balancing their residual energy, EBD will result in a slightly lower standard deviation than G-MSD. From Fig. 8(b), we observe that EBD with eAHC has the lowest standard deviation, which shows that eAHC can form groups of ELs such that the energy cost for an MS to visit each group is similar.

Fig. 9(a) shows the standard deviation of MSs' residual energy in each round. When $|\hat{L}| = 35$, the EBD method will not group ELs until the number of survived MSs is below 35 (in the 300th round). As EBD does not care residual energy of MSs but seeks to balance their energy consumption, the gap between maximum and minimum residual energy of MSs will increase as time goes by. That is why the standard deviation of EBD keeps growing. After the 300th round, most MSs exhaust their energy, so the standard deviation drops fast in EBD. On the contrary, our G-MSD algorithm considers residual energy of MSs by Eq. 5, so it always has the lowest standard deviation. Thus, G-MSD can better balancing the load of each MS. When $|\hat{L}| = 100$, the eAHC strategy can help EBD decrease its standard deviation, thereby extending system lifetime. On the other hand, G-MSD can keep a pretty low standard deviation as $|\hat{L}| = 100$, which shows its effectiveness on saving energy of MSs.

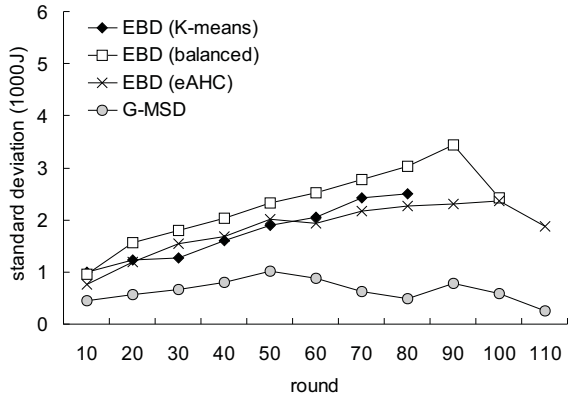
6.4 Effect of Grouping Strategies

We study the effect of different grouping strategies, including K-means, balanced, and eAHC, on the moving distance for each MS to visit ELs in a group. In the experiment, we set $|\hat{S}| = 20$ and iteratively increase $|\hat{L}|$ from 30 to 80. Fig. 10(a) gives the average length of the shortest path to visit all ELs in each group. Since the K-means strategy does not consider balancing the size of each group, some groups may contain ELs that are far away from each other. Thus, we need to find a longer path for the MS to visit ELs in such a group. On the contrary, the eAHC strategy iteratively merges two groups with the shortest inter-group distance, so it can form groups with smaller diameters, thereby reducing the average path length.

Fig. 10(b) gives the standard deviation of path length in each group. Obviously, when the standard deviation is smaller, it means that the sizes of formed groups (in terms of path lengths) are more balanced. In this way, MSs can have more fair loads on visiting ELs in their groups, which helps extend system lifetime. We can observe that both balanced and eAHC strategies result in much smaller standard deviations than the K-means strategy. Besides, these two strategies have similar values of standard deviation, which indicates that the balanced

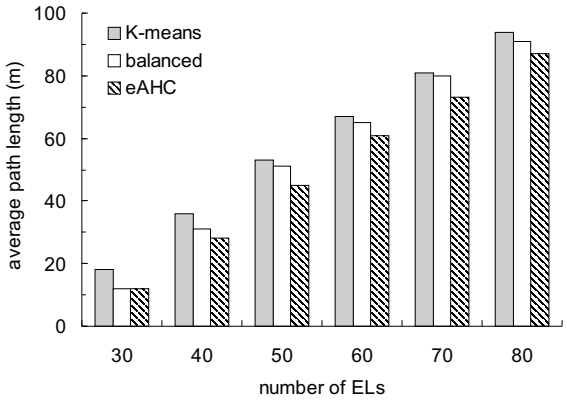


(a) 35 ELs

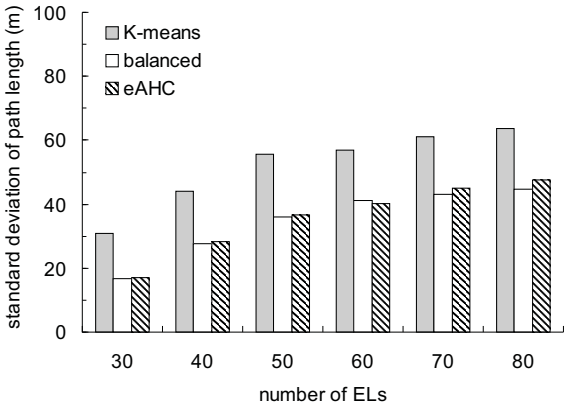


(b) 100 ELs

Fig. 9: Comparison on the standard deviation of residual energy of MSs.



(a) average



(b) standard deviation

Fig. 10: Effect of different grouping strategies on the length of a shortest path to visit all ELs in each group.

and eAHC strategies can form groups of ELs with more balanced sizes.

6.5 Effect of WCs

Then, we measure the effect of WCs on G-MSD's lifetime. To do so, we divide the sensing field into k equal subregions, and place one WC on the center of each region, where $k \in [1, 4]$. Fig. 11 shows system lifetime with 50 MSs when there are 40, 70, and 100 ELs. Obviously, the number of WCs has significant effect on lifetime when $|\hat{L}| = 40$. In this case, since there are more MSs than ELs, we can allow those MSs without dispatching jobs to visit WCs to replenish their energy, thereby substantially extend their lifetime. Thus, system lifetime greatly increases as the number of WCs grows. On the other hand, when there are more ELs than MSs (i.e., $|\hat{L}|$ is 70 or 100), placing more WCs cannot significantly prolong lifetime. The reason is that every MS has to be assigned with a dispatching job to visit ELs. Therefore, an MS may not have enough time to recharge its battery, so there is not much help in extending its lifetime by placing more WCs.

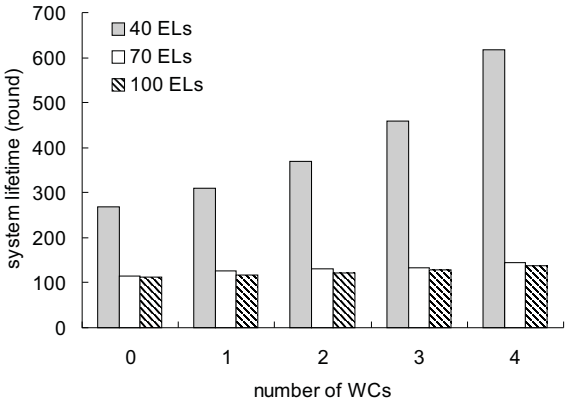


Fig. 11: Effect of the number of WCs on system lifetime.

6.6 Effect of Parameters

Recall that α decides the number of MSs that a WC can serve in each round. Table 4 shows its effect on system lifetime with 40 and 100 ELs, where we set $|\hat{C}| = 4$ and $|\hat{S}| = 50$. When $|\hat{L}| = 40$, lifetime keeps increasing in each method when we increase α from 1 to 5. Similarly, when $|\hat{L}| = 100$, lifetime also keeps increasing as we raise α from 1 to 4. However, when $\alpha \geq 5$, increasing α will not extend lifetime. The reason is that a WC can recharge no more than one MS at a time. As discussed in Section 5.2.2, the total recharging time of a WC is limited to $T^{\mathrm{C}}_{\mathrm{avg}}$. In this case, even though we relax the α constraint by setting $\alpha = \infty$, the energy replenishment scheme will not schedule more MSs to visit WCs. Thus, lifetime keeps stable when $\alpha \geq 5$ in the experiment.

In the G-MSD algorithm, we use a limit ε by Eq. 6 for each group of ELs to decide its candidate MSs and bid for an MS. Eq. 6 relies on a coefficient β to adjust the value of ε, so we evaluate its effect on system lifetime by G-MSD in Table 5, where $|\hat{S}| = 50$ and $|\hat{C}| = 0$. When $|\hat{L}| = 40$, we can maximize lifetime by setting $\beta = 20$. Otherwise, the peak value of lifetime occurs when $\beta = 15$. Thus, the suggested value of β is within $[15, 20]$ in this experiment.

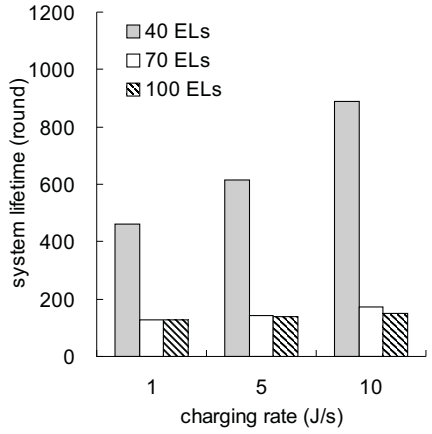
We then measure the effect of charging rate r on G-MSD's

TABLE 4: Effect of parameter α on system lifetime (in rounds).

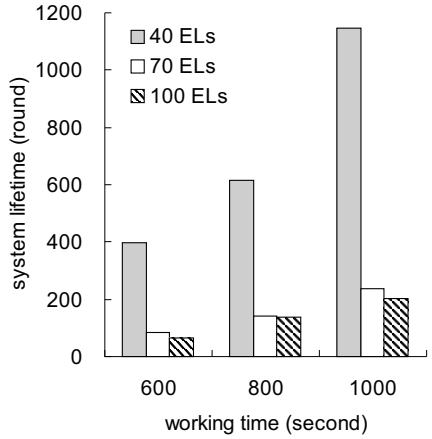
α value	1	2	3	4	5	6	∞
number of ELs: 40							
EBD (K-means)	417.2	418.1	425.5	426.7	429.3	427.1	427.8
EBD (balanced)	430.1	431.3	435.1	435.8	438.6	436.2	437.1
EBD (eAHC)	429.9	430.4	434.3	435.3	438.1	435.8	436.2
G-MSD	600.6	601.3	608.9	612.1	613.8	614.0	615.1
number of ELs: 100							
EBD (K-means)	81.9	89.4	94.5	94.9	95.3	94.3	94.1
EBD (balanced)	114.2	118.8	121.7	122.2	121.4	121.0	120.7
EBD (eAHC)	131.4	132.9	135.0	134.7	133.8	133.7	133.6
G-MSD	130.7	135.1	136.3	136.4	137.1	135.7	135.2

TABLE 5: Effect of parameter β on system lifetime (in rounds).

β value	5	10	15	20	25	30
30 ELs	465.7	472.2	472.5	469.7	468.2	466.3
40 ELs	258.8	268.2	271.7	272.0	270.6	270.3
50 ELs	114.2	116.7	123.4	123.3	121.7	121.5
60 ELs	106.1	108.9	111.6	111.4	108.7	106.1
70 ELs	110.1	112.7	115.0	113.4	109.8	107.5
80 ELs	110.9	113.8	115.7	112.8	109.7	106.3



(a) charging rate r



(b) working time $T_{\max}$

Fig. 12: Effect of charging rate and working time on system lifetime of G-MSD.

lifetime with 50 MSs. Fig. 12(a) gives the simulation result, where we set r to 1, 5, and 10 J/s. In general, a larger r value means that each WC can recharge a battery in a higher speed. Thus, when an MS calls at a WC, it can be replenished with more energy and extend lifetime accordingly. That is why the charging rate has significant impact when $|\hat{L}| < |\hat{S}|$ (i.e., 40 ELs), as there is a good possibility for MSs to visit WCs for recharging. When there are more ELs than MSs, each MS may need to visit multiple ELs and have not enough time to visit a WC. In this case, increasing r has less effect on lifetime.

Finally, we evaluate the working time $T_{\max}$ on G-MSD's lifetime, where there are also 50 MSs. Fig. 12(b) shows the result, where we set $T_{\max}$ to 600, 800, and 1000 seconds. Since $T_{\max}$ decides the length of a round, a larger $T_{\max}$ value implies that the frequency of event occurrence will decrease. In this case, each MS has ample time to finish its dispatching job and also visit a WC to recharge its battery. Thus, we can extend lifetime by increasing $T_{\max}$. Such a phenomenon is more obvious when there are 40 ELs, since a part of MSs will not be assigned with any job and have more time for WCs to recharge their batteries.

7 Conclusion

Introducing mobility to a WSN gives it flexibility to conduct various missions like event analysis. This paper aims at the multi-round sensor dispatch problem in a hybrid WSN with WCs, whose objective is to assign MSs to visit ELs and WCs to maximize system lifetime. We propose the G-MSD algorithm to solve this NP-hard problem. In G-MSD, we group ELs to improve dispatching efficiency and let groups bid for MSs to save and balance their energy costs. MSs are then scheduled to call at WCs on their ways to visit ELs, so as to increase recharging efficiency. Through simulations, we show that G-MSD extends more than 40% of system lifetime than the EBD method when MSs are more than ELs. Besides, the grouping strategy has a great impact on performance, where the eAHC strategy can cluster ELs into smaller groups, which helps reduce and balance energy costs of MSs. Moreover, we also show that G-MSD can further extend system lifetime by increasing the charging rate of WCs and the working time of MSs.

References

[1] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: a survey," *IEEE Comm. Surveys & Tutorials*, vol. 18, no. 1, pp. 553–576, 2016.

[2] P. Tokekar, J.V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[3] Y.C. Wang and C.C. Yang, "3S-cart: a lightweight, interactive sensor-based cart for smart shopping in supermarkets," *IEEE Sensors J.*, vol. 16, no. 17, pp. 6774–6781, 2016.

[4] C.A. Tokognon, B. Gao, G.Y. Tian, and Y. Yan, "Structural health monitoring framework based on Internet of things: a survey," *IEEE Internet of Things J.*, vol. 4, no. 3, pp. 619–635, 2017.

[5] Y.C. Wang and G.W. Chen, "Efficient data gathering and estimation for metropolitan air quality monitoring by using vehicular sensor networks," *IEEE Trans. Vehicular Technology*, vol. 66, no. 8, pp. 7234–7248, 2017.

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

[7] J. Rezazadeh, M. Moradi, and A.S. Ismail, "Mobile wireles sensor networks overview," *Int'l J. Computer Comm. and Networks*, vol. 2, no. 1, pp. 17–22, 2012.

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

[9] Y. Mei, Y.H. Lu, Y.C. Hu, and C.S.G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Trans. Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[10] X. Lu, P. Wang, D. Niyato, D.I. Kim, and Z. Han, "Wireless charging technologies: fundamentals, standards, and network applications," *IEEE Comm. Surveys & Tutorials*, vol. 18, no. 2, pp. 1413–1452, 2016.

[11] G. Jiang, S.K. Lam, Y. Sun, L. Tu, and J. Wu, "Joint charging tour planning and depot positioning for wireless sensor networks using mobile chargers," *IEEE/ACM Trans. Networking*, vol. 25, no. 4, pp. 2250–2266, 2017.

[12] D. Arivudainambi and S. Balaji, "Optimal placement of wireless chargers in rechargeable sensor networks," *IEEE Sensors J.*, vol. 18, no. 10, pp. 4212–4222, 2018.

[13] H.M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Trans. Control Systems Technology*, vol. 23, no. 1, pp. 52–63, 2015.

[14] K. Pereida, M.K. Helwa, and A.P. Schoellig, "Data-efficient multirobot, multitask transfer learning for trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1260–1267, 2018.

[15] Y. Wang and C.W. Silva, "Sequential Q-learning with Kalman filtering for multirobot cooperative transportation," *IEEE/ASME Trans. Mechatronics*, vol. 15, no. 2, pp. 261–268, 2010.

[16] Y. Wang, L. Cheng, Z.G. Hou, J. Yu, and M. Tan, "Optimal formation of multirobot systems based on a recurrent neural network," *IEEE Trans. Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 322–333, 2016.

[17] H. Hatime, R. Pendse, and J.M. Watkins, "Comparative study of task allocation strategies in multirobot systems," *IEEE Sensors J.*, vol. 13, no. 1, pp. 253–262, 2013.

[18] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, 2015.

[19] Y. Cui, X. Wu, J. Song, and H. Ma, "A dynamic task equilibrium allocation algorithm based on combinatorial auctions," *Proc. Int'l Conf. Intelligent Human-Machine Systems and Cybernetics*, 2016, pp. 530–533.

[20] Y. Zou and K. Chakrabarty, "Distributed mobility management for target tracking in mobile sensor networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 8, pp. 872–887, 2007.

[21] R. Tan, G. Xing, J. Wang, and H.C. So, "Exploiting reactive mobility for collaborative target detection in wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 3, pp. 317–332, 2010.

[22] J. Hu, L. Xie, and C. Zhang, "Energy-based multiple target localization and pursuit in mobile sensor networks," *IEEE Trans. Instrumentation and Measurement*, vol. 61, no. 1, pp. 212–220, 2012.

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

[24] X. Wang and S. Wang, "Hierarchical deployment optimization for wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 10, no. 7, pp. 1028–1041, 2011.

[25] Y.C. Wang, C.C. Hu, and Y.C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Trans. Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.

[26] Y.C. Wang and Y.C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–1294, 2008.

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

[28] A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 65–84, 2006.

[29] G. Wang, G. Cao, P. Berman, and T.F.L. Porta, "Bidding protocols for deploying mobile sensors," *IEEE Trans. Mobile Computing*, vol. 6, no. 5, pp. 515–528, 2007.

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

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

[32] S. Zhang, Z. Qian, J. Wu, F. Kong, and S. Lu, "Wireless charger placement and power allocation for maximizing charging quality," *IEEE Trans. Mobile Computing*, vol. 17, no. 6, pp. 1483–1496, 2018.

[33] P. Guo, X. Liu, S. Tang, and J. Cao, "Concurrently wireless charging sensor networks with efficient scheduling," *IEEE Trans. Mobile Computing*, vol. 16, no. 9, pp. 2450–2463, 2017.

[34] W. Liang, Z. Xu, W. Xu, J. Shi, G. Mao, and S.K. Das, "Approximation algorithms for charging reward maximization in rechargeable sensor networks via a mobile charger," *IEEE/ACM Trans. Networking*, vol. 25, no. 5, pp. 3161–3174, 2017.

[35] T. Liu, B. Wu, H. Wu, and J. Peng, "Low-cost collaborative mobile charging for large-scale wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 16, no. 8, pp. 2213–2227, 2017.

[36] P. Zhong, Y.T. Li, W.R. Liu, G.H. Duan, Y.W. Chen, and N. Xiong, "Joint mobile data collection and wireless energy transfer in wireless rechargeable sensor networks," *Sensors*, vol. 17, no. 8, pp. 1–23, 2017.

[37] S. Guo, C. Wang, and Y. Yang, "Joint mobile data gathering and energy provisioning in wireless rechargeable sensor networks," *IEEE Trans. Mobile Computing*, vol. 13, no. 12, pp. 2836–2852, 2014.

[38] Y. Zhang, S. He, and J. Chen, "Near optimal data gathering in rechargeable sensor networks with a mobile sink," *IEEE Trans. Mobile Computing*, vol. 16, no. 6, pp. 1718–1729, 2017.

[39] R.M. Buehrer, H. Wymeersch, and R.M. Vaghefi, "Collaborative sensor network localization: algorithms and practical issues," *Proc. The IEEE*, vol. 106, no. 6, pp. 1089–1114, 2018.

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

[41] Y.C. Wang and S.J. Liu, "Minimum-cost deployment of adjustable readers to provide complete coverage of tags in RFID systems," *J. Systems and Software*, vol. 134, pp. 228–241, 2017.

[42] V.S. Borkar, V. Ejov, J.A. Filar, and G.T. Nguyen, *Hamiltonian Cycle Problem and Markov Chains*. Berlin: Springer, 2012.

[43] D. Defays, "An efficient algorithm for a complete link method," *The Computer J.*, vol. 20, no. 4, pp. 364–366, 1977.

[44] M.D. Apperley and M.M. Alahmari, "Tracking battery state-of-charge in a continuous use off-grid electricity system," University of Waikato, Department of Computer Science, Tech. Rep., 04 2013.

[45] J.M. Soares, I. Navarro, and A. Martinoli, "The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox," *Proc. Iberian Robotics Conf.*, 2015, pp. 767–781.