A Two-Phase Dispatch Heuristic to Schedule the Movement of Multi-Attribute Mobile Sensors in a Hybrid Wireless Sensor Network

You-Chiun Wang

Abstract—The paper considers a hybrid wireless sensor network with static and mobile sensors, where each static sensor can detect only one attribute of event while a mobile sensor can analyze multiple attributes of events. Static sensors monitor the environment and report where events appear. Mobile sensors then move to these event locations to conduct more in-depth analysis. A critical issue is how to schedule the traveling paths of mobile sensors so as to extend their lifetime. We formulate this issue as a *multi-round multi-attribute sensor dispatch problem* and prove it to be NP-complete. Then, we develop a two-phase dispatch heuristic that adopts the concepts of Pareto optimality and spanning-tree construction. Our heuristic allows arbitrary numbers of mobile sensors and event locations and tries to reduce and balance the energy consumption of mobile sensors in each round. Through simulations, we verify the effectiveness of our heuristic. The paper contributes in defining a new sensor dispatch problem and developing an energy-efficient solution to the problem.

Index Terms—energy saving, mobile sensor, mobility management, Pareto optimality, wireless sensor network.

1 Introduction

With the development of MEMS and robotic techniques, mobile sensors are realized by installing sensing devices on mobile platforms. Hybrid *wireless sensor networks (WSNs)*, which consist of such mobile sensors and conventional static sensors, open new frontiers of WSN research. Static sensors form a backbone network to support environmental monitoring. Mobile sensors have more powerful sensing and computing capabilities. They can move to specific locations to carry out missions such as replacing broken nodes or analyzing suspicious events. Adding mobility to a WSN significantly improves its ability and reduces the deployment and maintenance costs [1]. Applications using hybrid WSNs have been proposed in [2]–[4].

In this paper, we are interested in the scenario of dispatching "multi-attribute" mobile sensors to the locations of events appearing in the sensing field. Static sensors are responsible for reporting where suspicious events appear. Mobile sensors then move to these *event locations* to conduct in-depth analysis. Given a set of attributes $\mathcal{A}$, each event reported by static sensor(s) is associated with one attribute in $\mathcal{A}$. Mobile sensors are equipped with multiple sensing devices, so each of them can analyze multiple attributes of events. We thus call them *multi-attribute mobile (MAM) sensors*. However, MAM sensors may have different attributes. When an event of attribute $a_i \in \mathcal{A}$ is reported by static sensor(s), only the MAM sensors with attribute a_i can be dispatched to analyze that event.

The above scenario has two characteristics. First, MAM sensors are *heterogeneous*. Second, *concurrent events* may arise in the sensing field. In fact, [5] points out that network and device homogeneity is an unrealistic assumption in most practical applications. A number of research efforts also consider heterogeneous mobile sensors and concurrent events. For example, [6] considers two types of underwater vehicles equipped with different marine sensors. The first type of vehicles track the positions and trajectories of unknown underwater objects. Then, they pass the information to the second type of vehicles to classify such underwater objects. In [7], N heterogeneous sensors installed on mobile platforms are used to detect, track, and pursue M moving targets. This explicitly means that M concurrent events arise in the network. The work of [8] considers that a parent mobile sensor pilots a set of child mobile sensors to visually map an environment, where the parent sensor is equipped with a laser scanner while each child sensor has a color camera. These research efforts motivate us to consider the problem of dispatching MAM sensors in a hybrid WSN.

Because MAM sensors are battery-powered and moving energy is critical for them [9], we aim at path efficiency of MAM sensors to save their energy. Assume that events may be kept for a while after their occurrence. Since events may appear anytime and anywhere [10], we suggest dividing time into *rounds* and then scheduling the traveling paths of MAM sensors in a round-by-round manner. Doing it this way may have two advantages. First, we need not frequently calculate how to dispatch MAM sensors, so the computation overhead is reduced. Second, we can develop a sophisticated dispatch algorithm, rather than designate one MAM sensor to analyze every fresh event by a greedy (but inefficient) approach.

This paper proposes an *MAM sensor dispatch problem*, which determines how to assign MAM sensors to visit event locations in every round such that the system lifetime is maximum. Here, the *system lifetime* is defined by the number of rounds until some event locations cannot be visited by any MAM sensor with the correct attribute due to lack of energy. We prove that even if future event locations can be predicted, the MAM sensor dispatch problem is still NP-complete. To solve this problem, we develop a two-phase heuristic whose idea is to reduce and balance the moving energy spent by MAM sensors in every round. In particular, phase 1 adopts the Pareto optimal concept to calculate one-to-one assignments between

Y.-C. Wang is with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, 80424, Taiwan. E-mail: ycwang@cse.nsysu.edu.tw

MAM sensors and event locations. Phase 2 then proposes a spanning-tree construction algorithm for MAM sensors to visit unassigned event locations. Simulation results show that our heuristic can extend the system lifetime compared with other schemes.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 defines the MAM sensor dispatch problem and gives a divide-and-conquer solution. Section 4 proposes our two-phase dispatch heuristic. Experimental results are presented in Section 5. We conclude the paper and outline some future work in Section 6.

2 Related Work

The subject of node mobility in *mobile ad hoc networks (MANETs)*, *multi-robot systems (MRSs)*, and WSNs have been widely investigated. Below, we discuss mobility management in MANETs, task allocation in MRSs, and sensor maneuver in WSNs. Then, we study the variations of the sensor dispatch problem in the literature.

2.1 Mobility Management in MANETs

Mobility management has received substantial attention in MANETs. Many studies [12]–[14] aim at topology control or network communication under frequent node mobility, where nodes may move arbitrarily or following some models [15]. Unlike the assumption in these studies, node mobility in a hybrid WSN could be controllable and even coordinated [1]. In a *delay-tolerant network (DTN)*, nodes may sometimes be disconnected, so packets must be delivered passively by waiting for the opportunity to get connected to the destination through node mobility. DTN can be viewed as a special case of MANET and some research efforts [16]–[18] investigate how to modify the nodes' trajectories to increase such an opportunity. Nevertheless, they aim at the communication issue, which is different from the objective of this paper.

2.2 Task Allocation in MRSs

MRS is one research topic in robots, which adopts multiple cooperative robots to accomplish a specific task in the uncertain environment [19]. To achieve this goal, several studies [20]–[22] develop *multi-agent reinforcement learning schemes* to train robots to learn the mappings between their statuses and actions. Obviously, these studies have different objectives from our paper.

Multi-robot task allocation, on the other hand, considers assigning multiple tasks to a team of mobile robots. Each task indicates a target location required to be visited by one robot, and the objective is to minimize the total cost (for example, the moving distance of all robots). This problem is shown to be NP-hard [23] and *market-based approaches* [24] are widely used to solve it. Specifically, robots compete in *auctions* for each task and the robot that wins the auction can visit the target location. Two types of auctions are considered. In *single-item auctions* [23], [25], [26], each robot submits a bid, and the one with the highest bid wins the auction. *Combinatorial auctions* [27], [28] are more complex, where multiple items are offered and each robot can bid on any combination of the subsets of these items. This allows robots to explicitly express the synergy among items. The multi-robot task allocation problem is similar to the sensor dispatch problem, and using auctions allows robots to bid for the target locations in a decentralized way. However, using auctions may not consider balancing the overall loads among robots, while our solution keeps the moving distances of MAM sensors as balanced as possible.

2.3 Sensor Maneuver in WSNs

Several research efforts investigate how to move mobile sensors to adjust the network topology for some purposes. For example, [29] moves sensors to approximate the distribution of events and maintain complete coverage of the sensing field. After identifying coverage holes, [30] discusses how to move sensors to fill these holes. Both [31] and [32] adopt virtual forces among sensors to make them evenly distribute over the sensing field. After computing the locations to place sensors, [33], [34] investigate how to dispatch sensors to these locations in an energy-efficient manner. This issue is similar to the sensor dispatch problem, but they consider minimizing the energy consumption of sensors in only one round.

Mobile sensors have also been considered to track moving objects. Given targets in a mobile WSN, [35] addresses how to move sensors to improve the quality of tracking a target, while avoid potential breakage of network connectivity and reduce the loss of sensing coverage due to sensor movement. In [36], static sensors identify the location of a tracking object and then multiple mobile sensors will move to that location to provide more in-depth detection. The accuracy of object detection is improved by the measurement of mobile sensors that have higher signal-to-noise ratios after movement. The work of [37] classifies mobile sensors into *leaders* and *followers*, where leaders are responsible for target localization and pursuit while followers take charge of maintaining network connectivity and team formation. However, the above studies aim at increasing the accuracy to detect objects and maintaining network connectivity, while our paper focuses on reducing the energy consumption of mobile sensors to extend their lifetime.

2.4 Variations of The Sensor Dispatch Problem

In the literature, some variations of the sensor dispatch problem have been proposed. The work in [38] considers dispatching sensors to some destinations in a one-to-one manner. When a sensor s_i is asked to visit a destination l_j, it does not directly move to l_j. Instead, s_i organizes a sequence $(s_i, s_{x_1}, s_{x_2}, \cdots, s_{x_k}, l_j)$ such that cascaded movements, $s_i \to s_{x_1}$, $s_{x_1} \to s_{x_2}$, $\cdots$, and $s_{x_k} \to l_j$, take place to balance the energy consumption among sensors. In [39], static sensors detecting events invite mobile sensors to visit them to provide more in-depth analysis. The mobile sensor which has a shorter moving distance and more energy, and whose leaving only causes a smaller uncovered hole, will be invited. The study in [40] dispatches mobile sensors to improve the sensing coverage of a hybrid WSN. Static sensors estimate the uncovered holes in their vicinity and then use the hole sizes to bid for mobile sensors. Given event locations in each round, [41] considers dispatching mobile sensors to visit these event locations in a round-by-round manner such that the number of rounds is maximum. The idea is to minimize the energy consumption of all mobile sensors while balancing their moving costs in every round. However, these studies assume that events are homogeneous (i.e., $|\mathcal{A}| = 1$). On the contrary, our work considers heterogeneous events and mobile sensors with multiple attributes. Therefore, a more challenging problem arises in the research of sensor dispatch.

3 MAM Sensor Dispatch Problem

We first formally define the MAM sensor dispatch problem and prove its NP-complete property, and then give an intuitive solution by using the divide-and-conquer concept.

3.1 Problem Definition

We are given a set of attributes $\mathcal{A}$ and a hybrid WSN consisting of static and MAM sensors. Sensors can use GPS (global positioning system) or some localization schemes [42] to obtain their own locations. Static sensors form a connected network that fully covers the sensing field. They cooperate to identify events of the same attribute that could appear anywhere in the sensing field[1]. Thus, each event reported by static sensor(s) is associated with exact one attribute in $\mathcal{A}$. In addition, we make no assumption on the distribution of event locations.

MAM sensors are equipped with multiple sensing devices and can visit event locations to perform more in-depth analysis. However, they may have different attributes. When an event of attribute $a_i \in \mathcal{A}$ is reported, we can only dispatch an MAM sensor with attribute a_i to analyze that event. In this case, the MAM sensor is said to have the *correct* attribute for the event. To guarantee that we can find MAM sensors to visit event locations, the union of all MAM sensors' attributes should be equal to $\mathcal{A}$. We assume that the moving speed of each MAM sensor and its energy cost to move a unit distance are both constants. In addition, the sensing field is considered as obstacle-free, so MAM sensors can directly move to their destinations via beelines[2].

For computation efficiency, we divide time into repetitive *rounds*. Each round is further divided into three periods:

- **Reporting period:** Static sensors report the events that have been detected but not yet processed in the period. Its length could depend on the frequency of event occurrence. This frequency can be learned by the statistics of event occurrence in previous rounds and used to adjust the length of the next reporting period.
- **Commanding period:** The sink then computes how to assign MAM sensors to visit the events received in the reporting period, and transmits the assignment to the corresponding MAM sensors in the commanding period. Obviously, its length depends on the network diameter.
- **Dispatching period:** In this period, MAM sensors move to their assigned event locations and perform analysis accordingly. The length of the dispatching period should include both the moving time of MAM sensors and the time spent to analyze events. It can be calculated by the sink in advance[3] and announced in the previous commanding period. In case that some MAM sensors are assigned with too many event locations or they have to spend a lot of time to analyze certain events, resulting in the length of the current round exceeding a maximum threshold, the sink can delete some event locations and put them in the next round for scheduling.

1. Refer to [39] for possible solutions.
2. Some studies [33], [43] discuss how to move sensors or robots in a sensing field with obstacles. They can help relax this assumption.
3. This can be achieved because the sink knows the locations and attributes of events from the information collected in the reporting period. Besides, the time spent to analyze events depends on the hardware capability and can be known beforehand.

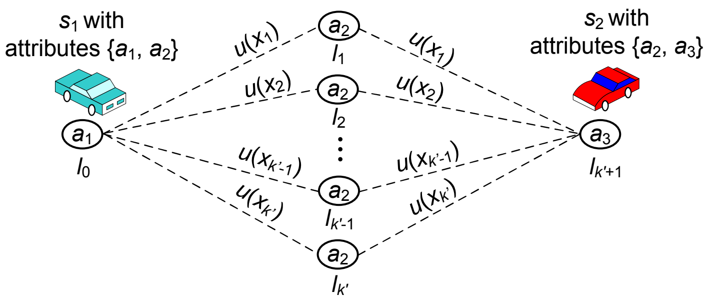


Fig. 1: An instance used to reduce the partition problem to the MSDD problem.

Our discussion aims at how to dispatch MAM sensors in each round. Specifically, given a set of m MAM sensors $\mathcal{S} = \{s_1, s_2, \cdots, s_m\}$ and a set of n event locations $\mathcal{L} = \{l_1, l_2, \cdots, l_n\}$ in a round, the objective is to assign each MAM sensor $s_i \in \mathcal{S}$ a *dispatch schedule* $\mathcal{D}_i$ that contains a sequence of event locations. Obviously, the union of event locations in all dispatch schedules should be equal to $\mathcal{L}$. Let $\mathcal{D}_i[j]$ be the jth location in $\mathcal{D}_i$ and e_i be the current energy of s_i. To complete s_i's dispatch schedule, the consumed energy is formulated as follows:

$$f(\mathcal{D}_i) = e_{\text{cost}} \times \left(d(s_i, \mathcal{D}_i[1]) + \sum_{j=1}^{|\mathcal{D}_i|-1} d(\mathcal{D}_i[j], \mathcal{D}_i[j+1]) \right),$$

where e_{cost} is the energy cost for an MAM sensor to move one unit distance, $|\mathcal{D}_i|$ is the number of event locations in $\mathcal{D}_i$, and $d(\cdot,\cdot)$ denotes the distance between two locations. Obviously, any dispatch schedule of s_i should satisfy that $e_i \geq f(\mathcal{D}_i)$. In addition, we are given s_i's initial energy e_i^{init} in round zero. Considering that MAM sensors are not rechargeable, our goal is to schedule their traveling paths such that the system lifetime is maximum.

Below, we prove that the MAM sensor dispatch problem is NP-complete even if the event locations in the future rounds are known in advance. To do so, the MAM sensor dispatch problem is formulated as a decision problem in Definition 1.

Definition 1. Given a set of MAM sensors $\mathcal{S}$ and a sequence of k event location sets, the MAM sensor dispatch decision (MSDD) problem determines whether there exists a feasible k-round schedule for $\mathcal{S}$ (in other words, the system lifetime has exact k rounds).

Theorem 1. The MSDD problem is NP-complete.

Proof: We first prove that the MSDD problem belongs to NP. Given an MSDD problem instance and a k-round schedule for $\mathcal{S}$, we can verify whether the schedule is valid or not in polynomial time. Therefore, the part is proved.

Then, we reduce an NP-complete problem, the *partition problem* [44], to the MSDD problem. Given a finite set $\mathcal{X}$ where each element $x_i \in \mathcal{X}$ is assigned with a value $u(x_i)$, the partition problem determines whether $\mathcal{X}$ can be divided into two subsets such that the sums of their values are equal.

Let $k = 2k'$ and $\mathcal{X} = \{x_1, x_2, \cdots, x_{k'}\}$ be an instance of the partition problem. We then construct an MSDD problem instance shown in Fig. 1. In the beginning, two MAM sensors s_1 and s_2 are located at l_0 and $l_{k'+1}$, respectively. Each of them has initial energy of $\sum_{i=1}^{k'} u(x_i)$. Let $\mathcal{A} = \{a_1, a_2, a_3\}$. MAM sensors s_1 and s_2 have attributes $\{a_1, a_2\}$ and $\{a_2, a_3\}$, respectively. For each $x_i \in \mathcal{X}$, we construct a location l_i such that the energy required to move from both l_0 and $l_{k'+1}$ to l_i is $u(x_i)$. Then, let us consider a problem instance with $2k'(=k)$ rounds, where in every $(2i-1)$th round one event of attribute a_2 appears at l_i, and in every $(2i)$th round one event of attribute a_1 appears at l_0 and one event of attribute a_3 appears at $l_{k'+1}$, $i = 1..k'$. Obviously, the construction of this problem instance requires only polynomial time. We then prove that $\mathcal{X}$ has a solution if and only if the MSDD problem has a feasible k-round schedule.

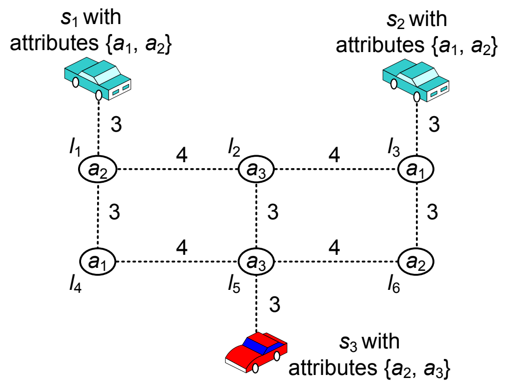
Suppose that there exists a feasible k-round schedule to the MSDD problem. The schedule must dispatch one MAM sensor to an event location in every odd round and then dispatch the same MAM sensor back to its original location in the subsequent even round. In other words, either s_1 or s_2 will move to l_i and move back in the $(2i-1)$th and the $(2i)$th rounds, $i = 1..k'$, respectively. Therefore, the overall energy consumption of s_1 and s_2 is $2 \cdot \sum_{i=1}^{k'} u(x_i)$. Because both s_1 and s_2 have initial energy of $\sum_{i=1}^{k'} u(x_i)$, each of them will exhaust its energy after $2k'(=k)$ rounds. Thus, s_1 and s_2 must have moved the same distance. Therefore, the sets of event locations visited by s_1 and s_2 all constitute a solution to the partition problem. Therefore, we prove the *if* part.

Conversely, suppose that subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ are a solution to the partition problem. Then, in every $(2i-1)$th round, $i = 1..k'$, we dispatch s_1 (respectively, s_2) to visit l_i when $\mathcal{X}_1$ (respectively, $\mathcal{X}_2$) contains an element with value $u(x_i)$, and move it back in the subsequent $(2i)$th round. In this case, both s_1 and s_2 must move the same distance and thus drain themselves of energy (that is, $\sum_{i=1}^{k'} u(x_i)$) after $2k'(=k)$ rounds. The above k-round schedule must be feasible to the MSDD problem, thereby proving the *only if* part. □

Remark 1. Sometimes, the same phenomenon may cause the trigger of different attributes of static sensors, which could be interpreted as the occurrence of multiple attributes of events. For example, a fire may cause a high temperature and emit several types of gases in the air. This case is inevitable since static sensors will report what they detect to the sink. One possible solution is to "define" events at the sink based on the reports from static sensors. Using the above example, the sink may define a "fire event" if static sensors in the same small region report both a high temperature and high CO_2 concentration. In this case, the sink can dispatch just one MAM sensor (rather than multiple ones) to visit the location of that fire event. This can avoid potential waste of energy. How to define events depends on the requirements of an application, and this issue is out of the paper's scope.

3.2 Divide-and-conquer (D&C) Solution

At first glance, one may intuitively adopt a *divide-and-conquer (D&C) scheme* by handling each attribute of event locations in a separate manner. Specifically, event locations are first clustered into groups based on their attributes. For each attribute $a_i \in \mathcal{A}$ of group, say, g_i, we assign the MAM sensors with attribute a_i, say, $\mathcal{S}_i$ to visit the event locations in g_i. To let all MAM sensors in $\mathcal{S}_i$ consume the least amount of energy, we draw a *complete* weighted bipartite graph based on g_i and $\mathcal{S}_i$, and then adopt the Hungarian algorithm [45] to calculate a minimum-weight maximum matching to schedule the MAM sensors in $\mathcal{S}_i$. This procedure is repeated until all groups are handled. Then, each MAM sensor adopts a *TSP (traveling salesman problem)* heuristic to visit all of its assigned event locations.

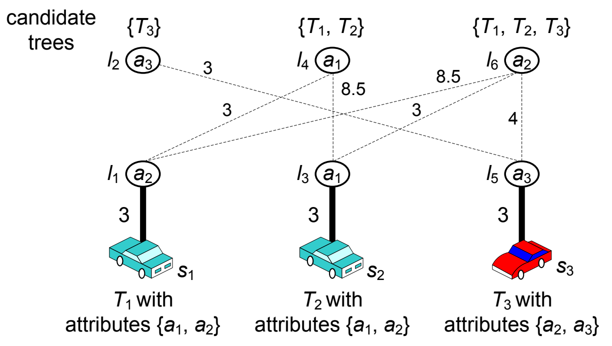


s1 with attributes {a1, a2}
s2 with attributes {a1, a2}
l1 a2
l2 a3
l3 a1
l4 a1
l5 a3
l6 a2
3
4
s3 with attributes {a2, a3}

(a)

	s_1	s_2	s_3	l_1	l_2	l_3	l_4	l_5	l_6
s_1	–	–	–	3	5	8.5	6	7.2	10
s_2	–	–	–	8.5	5	3	10	7.2	6
s_3	–	–	–	7.2	6	7.2	5	3	5
l_1	3	8.5	7.2	–	4	8	3	5	8.5
l_2	5	5	6	4	–	4	5	3	5
l_3	8.5	3	7.2	8	4	–	8.5	5	3
l_4	6	10	5	3	5	8.5	–	4	8
l_5	7.2	7.2	3	5	3	5	4	–	4
l_6	10	6	5	8.5	5	3	8	4	–

(b)



candidate trees
{T3}
{T1, T2}
{T1, T2, T3}
l2 a3
l4 a1
l6 a2
3
8.5
4
l1 a2
l3 a1
l5 a3
s1
s2
s3
T1 with attributes {a1, a2}
T2 with attributes {a1, a2}
T3 with attributes {a2, a3}

(c)

Fig. 2: An example of dispatching MAM sensors. (a) The distribution of MAM sensors and event locations. (b) The distance between any two locations, where the symbol '–' means that either the distance is zero or we do not care the distance. (c) Execution of the spanning-tree construction algorithm in phase 2.

Take an example in Fig. 2(a), where $\mathcal{A} = \{a_1, a_2, a_3\}$. There are six event locations in $\mathcal{L}$, l_1, l_2, l_3, l_4, l_5, and l_6 with attributes a_2, a_3, a_1, a_1, a_3, and a_2, respectively. In addition, there are three MAM sensors s_1, s_2, and s_3 in $\mathcal{S}$, which have attributes $\{a_1, a_2\}$, $\{a_1, a_2\}$, and $\{a_2, a_3\}$, respectively. Fig. 2(b) gives the distance between any two locations. Based on the above D&C scheme, we first handle the event locations with attribute a_1. In this case, we have $g_1 = \{l_3, l_4\}$ and $\mathcal{S}_1 = \{s_1, s_2\}$ and thus calculate two assignments: $s_1 \to l_4$ and $s_2 \to l_3$. For attribute a_2, we obtain that $g_2 = \{l_1, l_6\}$ and $\mathcal{S}_2 = \{s_1, s_2, s_3\}$. Thus, two assignments, $s_1 \to l_1$ and $s_3 \to l_6$, can be derived. Similarly, for attribute a_3, we have $g_3 = \{l_2, l_5\}$. Since $\mathcal{S}_3$ contains only s_3, we have to assign s_3 to visit both l_2 and l_5. Finally, by adopting a TSP heuristic, we schedule the moving paths of all MAM sensors as follows: $s_1 \to l_1 \to l_4$, $s_2 \to l_3$, and $s_3 \to l_5 \to l_2 \to l_6$, with the energy consumption of 6, 3, and 11, respectively.

The D&C scheme has two drawbacks. First, the overall

TABLE 1: Summary of notations.

Notation	Definition
$\mathcal{S}$	The set of MAM sensors, where $\lvert\mathcal{S}\rvert = m$
$\mathcal{L}$	The set of event locations in a round, where $\lvert\mathcal{L}\rvert = n$
$\mathcal{A}$	The set of attributes
$\mathcal{D}_i$	The dispatch schedule of an MAM sensor s_i, which contains a sequence of event locations to be visited
e_{cost}	The energy cost for each MAM sensor to move one unit distance
e_i^{init}	The initial energy of an MAM sensor s_i in round zero
$w(s_i, l_j)$	The energy consumption for an MAM sensor s_i to move from it current location to an event location l_j
$>_p$	Two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ are said to $\mathcal{M}_1 >_p \mathcal{M}_2$ if no MAM sensors prefer $\mathcal{M}_2$ to $\mathcal{M}_1$, and some MAM sensors prefer $\mathcal{M}_1$ to $\mathcal{M}_2$.
$\mathcal{L}_u$	The set of unassigned event locations after phase 1
$\mathcal{T}$	The set of candidate spanning trees in phase 2

moving energy of MAM sensors is not reduced, even though we try to minimize the moving costs of MAM sensors for each attribute of group. The reason is that each MAM sensor has only a "narrow view" in each group. Optimizing the moving path *inside* every group cannot guarantee to optimize the moving path *among* groups. In fact, we will see a better scheduling strategy using the same example latter. Second, some MAM sensors (e.g., s_3) are asked to move in longer distances, causing them to exhaust energy quickly. This will burden the survivals with heavy loads in the following rounds, resulting in a vicious spiral [41]. Therefore, we have to develop a more sophisticated and efficient algorithm to solve the MAM sensor dispatch problem.

4 Two-Phase Dispatch Heuristic

We develop a heuristic whose idea is to reduce the moving costs of MAM sensors while keeping their energy consumption as balanced as possible in each round. Without loss of generality, we remove the MAM sensors without sufficient energy to move to any event location in $\mathcal{L}$ from $\mathcal{S}$. Our dispatch heuristic is composed of two phases. In phase 1, we adopt a maximum-matching algorithm in a weighted bipartite graph to assign event locations to MAM sensors. Since phase 1 assigns at most one event location to an MAM sensor, there could remain unassigned event locations in $\mathcal{L}$. In this case, phase 2 is invoked to handle them by a spanning-tree construction algorithm. Then, each MAM sensor adopts a TSP heuristic to visit all event locations in its dispatch schedule. Table 1 summarizes the notations used in our heuristic.

4.1 Phase 1: Finding A Maximum Pareto-Optimal Matching

Given $\mathcal{S}$ and $\mathcal{L}$, we construct a weighted bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{S} \cup \mathcal{L}, \mathcal{S} \times \mathcal{L})$. All MAM sensors and all event locations are converted into vertices. Edges only connect vertices between $\mathcal{S}$ and $\mathcal{L}$. For every $s_i \in \mathcal{S}$ and every $l_j \in \mathcal{L}$, there exists an edge (s_i, l_j) between them if and only if both s_i and l_j have the same attribute. The corresponding weight is then calculated by $w(s_i, l_j) = e_{\text{cost}} \times d(s_i, l_j)$. Therefore, the MAM sensor dispatch problem can be translated to the problem of finding a matching $\mathcal{M}$ from $\mathcal{G}$ such that

1) The number of matches in $\mathcal{M}$ is maximum.
2) Matching $\mathcal{M}$ is Pareto optimal.

Objective 1) is to maximize the number of event locations handled in this phase. On the other hand, since each edge weight takes the energy consumption as the metric, objective 2) is to reduce the overall energy spent by all MAM sensors and balance their energy costs. Below, we call a matching which satisfies both objectives a *maximum Pareto-optimal matching*.

To find a maximum matching $\mathcal{M}$ in objective 1), we adopt the Hopcroft-Karp algorithm [46], which iteratively increases the size of a partial matching by calculating augmenting paths. Specifically, a vertex $v_i \in \mathcal{V}$ is considered as *free* if $(v_i, v_j) \notin \mathcal{M}$ for all $(v_i, v_j) \in \mathcal{E}$. In addition, we call a path $\mathcal{P} = \{(v_1, v_2), (v_2, v_3), \cdots, (v_{i-1}, v_i)\}$ an *augmenting path* if both of its endpoints v_1 and v_i are free, and its edges alternatively appear in $\mathcal{E} - \mathcal{M}$ and $\mathcal{M}$. Suppose that $\mathcal{M}$ has a size of k (that is, $\mathcal{M}$ has k matches) and $\mathcal{P}$ is an augmenting path relative to $\mathcal{M}$. Then, the symmetric difference of the edges in $\mathcal{M}$ and the edges in $\mathcal{P}$ must form a new matching whose size is $k+1$. According to this observation, we adopt a three-step scheme to find the maximum matching $\mathcal{M}$:

1) Initially, let $\mathcal{M} = \emptyset$. We then arbitrarily select one edge from $\mathcal{E}$ and add it to $\mathcal{M}$.
2) From $\mathcal{M}$, we find its augmenting path $\mathcal{P}$, and generate a new matching $\mathcal{M}' = \mathcal{M} \oplus \mathcal{P}$, which contains the symmetric difference of the edges in $\mathcal{M}$ and the edges in $\mathcal{P}$. Then, we set $\mathcal{M} = \mathcal{M}'$.
3) Repeat step 2 until there is no augmenting path from $\mathcal{M}$.

Lemma 1 gives the time complexity of the above scheme.

Lemma 1. Finding a maximum matching $\mathcal{M}$ in $\mathcal{G}$ requires $O(mn\sqrt{m+n})$ time in the worst case.

Proof: According to the analysis in [46], it takes $O(\lvert\mathcal{E}\rvert \cdot \sqrt{\lvert\mathcal{V}\rvert})$ time to calculate the maximum matching $\mathcal{M}$ by using the augmenting-path solution, where $\lvert\mathcal{E}\rvert$ and $\lvert\mathcal{V}\rvert$ are the number of edges and the number of vertices in $\mathcal{G}$, respectively. The worst case occurs when $\mathcal{G}$ is a complete weighted bipartite graph. In this case, because

$$\lvert\mathcal{V}\rvert = \lvert\mathcal{S} \cup \mathcal{L}\rvert = \lvert\mathcal{S}\rvert + \lvert\mathcal{L}\rvert = m + n,$$
$$\lvert\mathcal{E}\rvert = \lvert\mathcal{S} \times \mathcal{L}\rvert = \lvert\mathcal{S}\rvert \times \lvert\mathcal{L}\rvert = mn,$$

finding $\mathcal{M}$ requires $O(\lvert\mathcal{E}\rvert \cdot \sqrt{\lvert\mathcal{V}\rvert}) = O(mn\sqrt{m+n})$ time. □

Then, to satisfy objective 2), we "exchange" some matches in $\mathcal{M}$ to make it become Pareto optimal. To do so, we first define the *preference* of an MAM sensor in relation to matchings in Definition 2.

Definition 2. Given two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$, an MAM sensor s_i is said to prefer $\mathcal{M}_1$ to $\mathcal{M}_2$ if either of the two conditions is satisfied:

- MAM sensor s_i is matched to one event location in $\mathcal{M}_1$ but not in $\mathcal{M}_2$.
- Suppose that s_i is matched to event locations l_j and l_k in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. We have $w(s_i, l_j) < w(s_i, l_k)$, so s_i spends less energy to reach l_j than to l_k.

However, if s_i is not matched in both $\mathcal{M}_1$ and $\mathcal{M}_2$, it is considered as no difference.

Let us denote by $\mathcal{M}_1 >_p \mathcal{M}_2$ when no MAM sensors prefer $\mathcal{M}_2$ to $\mathcal{M}_1$, and some MAM sensors prefer $\mathcal{M}_1$ to $\mathcal{M}_2$. Then, Definition 3 defines the Pareto optimality.

Definition 3. A matching $\mathcal{M}$ is Pareto optimal if and only if we cannot find another matching, say, $\mathcal{M}'$ such that $\mathcal{M}' >_p \mathcal{M}$.

By modifying the algorithm in [11], we conduct two checks successively to make $\mathcal{M}$ become Pareto optimal.

- **Trade-in-free check:** Suppose that an MAM sensor s_i is matched to an event location l_j in $\mathcal{M}$. If there exists an unmatched event location, say, l_k such that $w(s_i, l_j) > w(s_i, l_k)$, we replace the match (s_i, l_j) with a new match (s_i, l_k) in $\mathcal{M}$. When there are multiple candidates, we select the event location l_k such that $w(s_i, l_k)$ is minimum. The trade-in-free check is repeated until no such replacement can be conducted.
- **Coalition-free check:** Suppose that there exists a sequence of matches $(s_{i_1}, l_{j_1}), (s_{i_2}, l_{j_2}), \cdots, (s_{i_k}, l_{i_k})$ in $\mathcal{M}$ such that $w(s_{i_1}, l_{j_1}) > w(s_{i_1}, l_{j_2}), w(s_{i_2}, l_{j_2}) > w(s_{i_2}, l_{j_3}), \cdots, w(s_{i_{k-1}}, l_{j_{k-1}}) > w(s_{i_{k-1}}, l_{j_k})$, and $w(s_{i_k}, l_{j_k}) > w(s_{i_k}, l_{j_1})$. In this case, we remove matches (s_{i_1}, l_{j_1}), (s_{i_2}, l_{j_2}), $\cdots$, (s_{i_k}, l_{i_k}) from $\mathcal{M}$ and then add new matches (s_{i_1}, l_{j_2}), (s_{i_2}, l_{j_3}), $\cdots$, $(s_{i_{k-1}}, l_{j_k})$, and (s_{i_k}, l_{j_1}) to $\mathcal{M}$. The coalition-free check is repeated until no such sequence can be found.

In the trade-in-free check, although an MAM sensor has already matched one event location, it can still test if there are unmatched event locations closer than the current matched one. If so, the MAM sensor can change to visit the closest unmatched event location to save its moving energy. On the other hand, the coalition-free check allows MAM sensors to exchange their matched event locations to further reduce the overall energy consumption. In particular, if there exists a sequence of MAM sensors $(s_{i_1}, s_{i_2}, \cdots, s_{i_k})$ in $\mathcal{M}$ such that every MAM sensor s_{i_α} prefers the event location matched to $s_{i_{\alpha+1}}$ (i.e., $l_{j_{\alpha+1}}$) to its matched event location l_{j_α}, for $\alpha = 1..k-1$, and s_{i_k} prefers l_{j_1} (which is matched to s_{i_1}) to its matched event location l_{j_k}, a *coalition* $\{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$ occurs. In this case, every s_{i_α} and $s_{i_{\alpha+1}}$ in the coalition (including s_{i_k} and s_{i_1}) can exchange their matched event locations to reduce the overall moving energy. Notice that the number of matches in $\mathcal{M}$ does not change after passing the above two checks, so $\mathcal{M}$ is still a maximum matching. Theorem 2 proves the correctness of the two checks and Lemma 2 gives their computation complexity.

Theorem 2. A maximum matching $\mathcal{M}$ is Pareto optimal after conducting the trade-in-free and coalition-free checks.

Proof: Suppose that $\mathcal{M}$ is Pareto optimal. If $\mathcal{M}$ is not trade-in-free, there must be an MAM sensor s_i and an event location l_k such that s_i is matched to l_j in $\mathcal{M}$, l_k is not matched in $\mathcal{M}$, and s_i prefers l_k to l_j. Thus, we can derive another matching $\mathcal{M}' = (\mathcal{M} \backslash \{(s_i, l_j)\}) \cup \{(s_i, l_k)\}$, where we remove match (s_i, l_j) from $\mathcal{M}$ and then add match (s_i, l_k) to $\mathcal{M}$. Obviously, since $\mathcal{M}' >_p \mathcal{M}$, a contradiction occurs. On the other hand, if $\mathcal{M}$ has a coalition $\{s_{i_1}, s_{i_2}, \cdots, s_{i_k}\}$, we can derive another matching $\mathcal{M}'$ by exchanging the matches in the coalition. In this case, a contradiction occurs because $\mathcal{M}' >_p \mathcal{M}$. Therefore, we prove the *if* part.

Conversely, suppose that $\mathcal{M}$ is both trade-in-free and coalition-free but not Pareto optimal. Then, there must exist another matching $\mathcal{M}'$ such that $\mathcal{M}' >_p \mathcal{M}$. Let s_{i_1} be an MAM sensor matched in $\mathcal{M}$ and s_{i_1} prefers $\mathcal{M}'$ to $\mathcal{M}$. For ease of presentation, let us denote by $\mathcal{M}(s_{i_\alpha})$ and $\mathcal{M}'(s_{i_\alpha})$ the event locations that s_{i_α} is matched to in $\mathcal{M}$ and in $\mathcal{M}'$, respectively. Then, it follows that $\mathcal{M}'(s_{i_1})$ is matched in $\mathcal{M}$, say, to s_{i_2} (otherwise, $\mathcal{M}$ cannot be trade-in-free). In this case, we have $\mathcal{M}'(s_{i_2}) \neq \mathcal{M}(s_{i_2})$ and s_{i_2} must prefer $\mathcal{M}'$ to $\mathcal{M}$ (otherwise, $\mathcal{M}' >_p \mathcal{M}$ cannot be true). Applying the similar argument, $\mathcal{M}'(s_{i_2})$ is matched in $\mathcal{M}$, say, to s_{i_3} and s_{i_3} prefers $\mathcal{M}'$ to $\mathcal{M}$. Following the same way, we can find a sequence of MAM sensors $\{s_{i_1}, s_{i_2}, \cdots\}$ such that s_{i_α} prefers $\mathcal{M}(s_{i_{\alpha+1}})$ to $\mathcal{M}(s_{i_\alpha})$. Because $\mathcal{S}$ is finite, the above sequence of MAM sensors must form a cycle. This statement obviously contradicts the assumption that $\mathcal{M}$ is coalition-free, thereby proving the *only if* part. □

Lemma 2. It takes $O(mn \lg n)$ time in the worst case to perform the trade-in-free and coalition-free checks on a maximum matching $\mathcal{M}$.

Proof: To efficiently conduct the trade-in-free and coalition-free checks, we maintain a *preference list* for each MAM sensor $s_i \in \mathcal{S}$, which ranks all of the event locations acceptable to s_i (that is, s_i has the correct attributes for these event locations). When $\mathcal{G}$ is a *complete* weighted bipartite graph, each $s_i \in \mathcal{S}$ has to sort $|\mathcal{L}| = n$ event locations in its preference list, which spends $O(n \lg n)$ time. Thus, the worst-case time complexity to compute the preference lists of all MAM sensors is

$$|\mathcal{S}| \cdot O(n \lg n) = m \cdot O(n \lg n) = O(mn \lg n).$$

Besides, the length of all preference lists is $|\mathcal{S} \times \mathcal{L}| = mn$. According to [11], the repeated trade-in-free checks can be realized by searching all preference lists just once. Thus, it takes $O(mn)$ time for $\mathcal{M}$ to pass all trade-in-free checks. Similarly, the repeated coalition-free checks can be implemented by a one-time search of all preference lists. Thus, it also spends $O(mn)$ for $\mathcal{M}$ to pass all coalition-free checks. Therefore, the overall time complexity is

$$O(mn \lg n) + O(mn) + O(mn) = O(mn \lg n).$$

□

Let us use the example in Fig. 2(a) to demonstrate the above matching-finding algorithm. From $\mathcal{S} = \{s_1, s_2, s_3\}$ and $\mathcal{L} = \{l_1, l_2, l_3, l_4, l_5, l_6\}$, we construct a weighted bipartite graph and assign edge weights based on Fig. 2(b). Then, we select an edge, say, (s_1, l_1) and run the Hopcroft-Karp algorithm. By iteratively calculating the augmenting path, we eventually obtain a maximum matching $\mathcal{M} = \{(s_1, l_6), (s_2, l_4), (s_3, l_1)\}$. Then, we conduct the trade-in-free check on $\mathcal{M}$. Starting from s_3, we find an unmatched event location l_5 such that $w(s_3, l_5) = 3 < w(s_3, l_1) = 7.2$. Thus, we replace (s_3, l_1) with (s_3, l_5) in $\mathcal{M}$. Since l_1 becomes unassigned and $w(s_1, l_1) = 3 < w(s_1, l_6) = 10$, we thus replace (s_1, l_6) with (s_1, l_1) in $\mathcal{M}$. Similarly, (s_2, l_4) is replaced with (s_2, l_3) in $\mathcal{M}$. Therefore, we have a new maximum matching $\mathcal{M} = \{(s_1, l_1), (s_2, l_3), (s_3, l_5)\}$. Since there are no coalitions in $\mathcal{M}$, it can pass the coalition-free check and thus $\mathcal{M}$ is Pareto optimal.

Theorem 3 presents the time complexity of the matching-finding algorithm.

Theorem 3. Executing the matching-finding algorithm requires $O(mn\sqrt{m+n})$ time in the worst case.

Proof: It takes $O(mn)$ time to construct a complete weighted bipartite graph $\mathcal{G}$, because we need to assign a weight for each edge in $\mathcal{E} = \mathcal{S} \times \mathcal{L}$. According to Lemmas 1 and 2, finding a maximum Pareto-optimal matching $\mathcal{M}$ from $\mathcal{G}$ requires $O(mn\sqrt{m+n}) + O(mn \lg n)$ time. Therefore, the matching-finding algorithm spends time of

$$O(mn) + O(mn\sqrt{m+n}) + O(mn \lg n) = O(mn\sqrt{m+n}).$$

□

We then discuss the rationale in phase 1. Recall that the D&C scheme in Section 3.2 limits the view of each MAM sensor to only a group of event locations, leading some MAM sensors to move in longer distances. To overcome this weakness, we construct a weighted bipartite graph $\mathcal{G}$ that takes all event locations into consideration. Although the Hungarian algorithm is widely used to find a minimum-weight maximum matching in a bipartite graph, we do not adopt it in phase 1 due to two reasons:

- The Hungarian algorithm requires $O(K^3)$ time to find the matching, where $K = \max(m, n)$, while our matching-finding algorithm takes only $O(mn\sqrt{m+n})$ time, which is more computation-efficient.
- Pareto optimality can maximize people's satisfaction when they have different requirements in some economic issues. By translating the MAM sensor dispatch problem to the problem of finding a maximum Pareto-optimal matching $\mathcal{M}$ from $\mathcal{G}$, we could not only reduce the energy consumption of MAM sensors but also balance their moving costs because each MAM sensor is "satisfied" with its assigned event location in $\mathcal{M}$.

4.2 Phase 2: Constructing Multiple Spanning Trees

Phase 1 uses a one-to-one manner to assign MAM sensors to visit event locations. However, when MAM sensors are not enough to visit all event locations, some event locations will be left *unassigned*. Therefore, we propose a spanning-tree construction algorithm to handle this situation in phase 2.

Given $\mathcal{M}$ from phase 1, each of its matches is treated as a spanning tree rooted at the corresponding MAM sensor. Let $\mathcal{L}_u$ be the set of unassigned event locations after phase 1. Then, for each event location $l_j \in \mathcal{L}_u$, it can join a spanning tree if the root MAM sensor has the correct attribute. Let $\mathcal{T}$ be the set of candidate trees such that l_j can join, the goal is to select a tree t_i from $\mathcal{T}$ such that

1) The original tree weight of t_i is as small as possible.
2) When l_j joins t_i, the increase in its tree weight is minimum.

The first objective is to balance the moving costs among MAM sensors, while the second objective is to reduce the energy consumption of t_i's root MAM sensor.

We develop a spanning-tree construction algorithm to satisfy both objectives as follows:

1) For each match (s_i, l_j) in $\mathcal{M}$, we construct a spanning tree such that the root is s_i and the tree contains an edge (s_i, l_j).
2) Select one event location, say, l_j from $\mathcal{L}_u$. Let a_j be the attribute of l_j. We then calculate the set of candidate trees $\mathcal{T}$ whose root MAM sensors have attribute a_j. In case of $\mathcal{T} = \emptyset$, the algorithm is terminated since no MAM sensors have the correct attribute to handle l_j.
3) If $|\mathcal{T}| = 1$, l_j joins the only tree in $\mathcal{T}$ and we update its tree weight accordingly. Otherwise, we sort all trees in $\mathcal{T}$ based on their tree weights in an ascending order. Then, we pick the first $\lceil \delta \cdot |\mathcal{T}| \rceil$ trees from $\mathcal{T}$, where $0 < \delta \leq 1$. Among these trees, l_j joins the tree t_i such that the increase in t_i's tree weight is minimum. We then update t_i's tree weight accordingly.
4) Delete l_j from $\mathcal{L}_u$ as it has been assigned with an MAM sensor. We then repeat steps 2 and 3 until $\mathcal{L}_u = \emptyset$.

Let us continue the example in Fig. 2(a) to demonstrate the spanning-tree construction algorithm. Given the maximum Pareto-optimal matching $\mathcal{M} = \{(s_1, l_1), (s_2, l_3), (s_3, l_5)\}$ from phase 1, we construct a set of spanning trees T_1, T_2, and T_3 shown in Fig. 2(c), where their tree weights are all 3. Meanwhile, we have a set of unassigned event locations $\mathcal{L}_u = \{l_2, l_4, l_6\}$. Let $\delta = 0.6$. Starting from l_2, since its attribute is a_3, only T_3 can be its candidate tree. Thus, l_2 joins T_3 and T_3's tree weight becomes $3 + 3 = 6$. Then, for l_4, it has two candidate trees $\mathcal{T} = \{T_1, T_2\}$. Since $\lceil \delta \cdot |\mathcal{T}| \rceil = \lceil 0.6 \times 2 \rceil = 2$, l_4 has to choose between T_1 and T_2. Because T_1 can increase a smaller tree weight (i.e., 3) when l_4 joins it, comparing with that to T_2 (i.e., 8.5), we thus let l_4 join T_1. Similarly, among candidate trees T_1, T_2, and T_3, l_6 selects T_2 to join. Therefore, the final spanning trees are $T_1 = \{s_1, l_1, l_4\}$, $T_2 = \{s_2, l_3, l_6\}$, and $T_3 = \{s_3, l_5, l_2\}$.

Notice that phase 2 uses only the MAM sensors in $\mathcal{M}$ to grow the spanning trees. In other words, if an MAM sensor is not assigned with any event location in phase 1, it cannot participate in phase 2. We show this property in both Theorem 4 and Corollary 1.

Theorem 4. Every MAM sensor can be always assigned with an event location in phase 1 unless it does not have the correct attribute for any event location in $\mathcal{L}_u$.

Proof: We prove this theorem by considering two possible cases: $|\mathcal{S}| \geq |\mathcal{L}|$ and $|\mathcal{S}| < |\mathcal{L}|$.

Case of $|\mathcal{S}| \geq |\mathcal{L}|$: This case is divided into two subcases.

- If all event locations in $\mathcal{L}$ are assigned with MAM sensors in phase 1, then $\mathcal{L}_u = \emptyset$ and thus the theorem is proven.
- Otherwise, $\mathcal{L}_u$ must be nonempty. Since we have $|\mathcal{S}| \geq |\mathcal{L}|$, there must exist a subset $\mathcal{S}_u \subseteq \mathcal{S}$ of MAM sensors which are not assigned with any event location in phase 1. Then, we show that every MAM sensor in $\mathcal{S}_u$ does not have the correct attribute for any event location in $\mathcal{L}_u$ by contradiction. Suppose that an MAM sensor $s_i \in \mathcal{S}_u$ and an event location $l_j \in \mathcal{L}_u$ have the same attribute. Then, we can derive a new matching $\mathcal{M}' = \mathcal{M} \cup \{(s_i, l_j)\}$. Obviously, we have $\mathcal{M}' >_p \mathcal{M}$, which results in a contradiction since $\mathcal{M}$ is Pareto optimal. Therefore, the theorem is correct.

Case of $|\mathcal{S}| < |\mathcal{L}|$: In this case, we have $\mathcal{L}_u \neq \emptyset$. Similarly, the case is divided into two subcases.

- If $\mathcal{S}_u = \emptyset$, in the sense that every MAM sensor is assigned with one event location in phase 1, then the theorem is proven.
- Otherwise, we also show the correctness of the theorem by contradiction. Suppose that an MAM sensor $s_i \in \mathcal{S}_u$ has the correct attribute for an event location $l_j \in \mathcal{L}_u$. Because $\mathcal{M}$ must have $|\mathcal{S} - \mathcal{S}_u|$ matches, we can derive another matching $\mathcal{M}'$ by adding a new match (s_i, l_j) to $\mathcal{M}$. Notice that $\mathcal{M}'$ must exist since $|\mathcal{S} - \mathcal{S}_u| < |\mathcal{S}|$. In this case, $\mathcal{M}'$ has more matches than $\mathcal{M}$, which contradicts the assumption that $\mathcal{M}$ is maximum. Therefore, the theorem is proved.

□

Corollary 1. If an MAM sensor does not appear in $\mathcal{M}$, it cannot participate in phase 2.

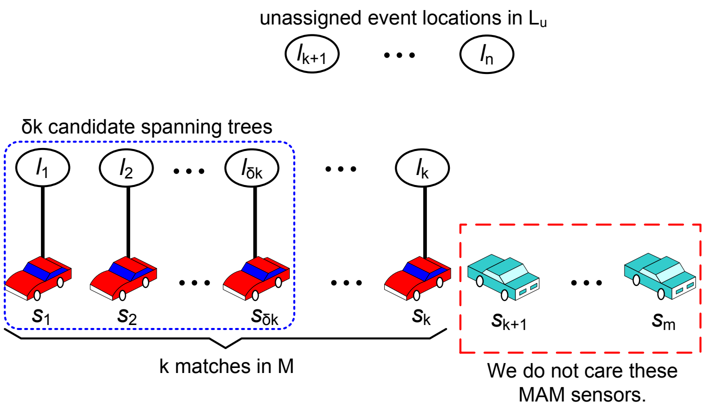


Fig. 3: Calculating the time complexity of the spanning-tree construction algorithm.

Proof: Based on Theorem 4, since this MAM sensor has no correct attribute for any event location in $\mathcal{L}_u$, it is not considered in phase 2. □

Corollary 1 is critical in analyzing the time complexity of the spanning-tree construction algorithm, which is presented in Theorem 5 and Corollary 2.

Theorem 5. Given matching $\mathcal{M}$ from phase 1, the spanning-tree construction algorithm requires $O(n \lg |\mathcal{M}|) + O(\delta|\mathcal{M}| \cdot (n - |\mathcal{M}|)) + O((n - |\mathcal{M}|)^2)$ time in the worst case.

Proof: Let $|\mathcal{M}| = k$. Without loss of generality, suppose that $\mathcal{M} = \{(s_1, l_1), (s_2, l_2), \cdots, (s_k, l_k)\}$. Based on Corollary 1, only the MAM sensors in $\mathcal{M}$ can participate in the spanning-tree construction algorithm. Thus, we have a set of k spanning trees rooted at $s_1, s_2, \cdots, s_k$ and $\mathcal{L}_u = \{l_{k+1}, l_{k+2}, \cdots, l_n\}$, as shown in Fig. 3. The worst case occurs when each event location in $\mathcal{L}_u$ can select all of these k trees to be its candidates. Then, we sort these k trees using their tree weights, which takes $O(k \lg k)$ time, and analyze how the spanning-tree construction algorithm reacts to each event location in $\mathcal{L}_u$:

- For l_{k+1}, it has δk trees to select according to step 3 of the algorithm[4]. In this case, there are totally $2\delta k$ edges to be tested (that is, edges (l_{k+1}, s_1), (l_{k+1}, s_2), $\cdots$, $(l_{k+1}, s_{\delta k})$, (l_{k+1}, l_1), (l_{k+1}, l_2), $\cdots$, $(l_{k+1}, l_{\delta k})$, as shown in Fig. 3). Among these $2\delta k$ edges, l_{k+1} selects the edge with the minimum weight and join the corresponding tree, say, t_i. Then, we update t_i's tree weight and sort all k trees again. Since all trees (except t_i) have been sorted, we can use a binary search to insert t_i in the correct position of the sorting list. This requires $O(\lg k)$ time.
- For l_{k+2}, it also has δk trees to select. In this case, there are totally $2\delta k + 1$ edges to be tested (since l_{k+1} has joined one tree, there would be a new edge (l_{k+2}, l_{k+1}) to be tested). Similarly, after updating the tree weight of the selected tree, it takes $O(\lg k)$ time to sort k trees using the binary search.
- Following the same way, for l_{k+j}, it needs to test $2\delta k + (j-1)$ edges. Then, sorting k trees spends $O(\lg k)$ time.
- Finally, for l_n, it must test $2\delta k + (n-k-1)$ edges. However, we need not sort trees in the last iteration.

4. We ignore the ceiling function here for ease of presentation.

Since there are $(n-k)$ event locations in $\mathcal{L}_u$, we sum up the above calculations as follows:

$$\sum_{j=1}^{n-k} 2\delta k + (j-1) + \sum_{j=1}^{n-k-1} O(\lg k) \tag{1}$$
$$= 2\delta k(n-k) + \frac{(n-k-1)(n-k)}{2} + (n-k-1) \cdot O(\lg k).$$

Here, the first term and the second term in Eq. (1) indicate the time spent to find the edge with the minimum weight for each event location and to sort k trees in every iteration (except the last one), respectively. By adding the big-O notation to Eq. (1), we obtain

$$O(\delta k(n-k)) + O((n-k)^2) + O((n-k) \cdot \lg k).$$

Therefore, the overall time complexity of the spanning-tree construction algorithm is

$$O(k \lg k) + O(\delta k(n-k)) + O((n-k)^2) + O((n-k) \cdot \lg k)$$
$$= O(n \lg k) + O(\delta k(n-k)) + O((n-k)^2). \tag{2}$$

By replacing k with $|\mathcal{M}|$ in Eq. (2), we obtain

$$O(n \lg |\mathcal{M}|) + O(\delta|\mathcal{M}| \cdot (n - |\mathcal{M}|)) + O((n - |\mathcal{M}|)^2).$$

□

Corollary 2. When $n \gg |\mathcal{M}|$, the time complexity of the spanning-tree construction algorithm is $O(n^2)$.

When there are sufficient MAM sensors, our two-phase dispatch heuristic can properly assign MAM sensors to visit all event locations in $\mathcal{L}$. Then, for every MAM sensor with a nonempty dispatch schedule, it can adopt any TSP heuristic [47] to efficiently reach each of its assigned event locations[5]. Let us complete the example in Fig. 2(a). By using a TSP heuristic on the spanning trees $T_1 = \{s_1, l_1, l_4\}$, $T_2 = \{s_2, l_3, l_6\}$, and $T_3 = \{s_3, l_5, l_2\}$ calculated in phase 2, we can schedule the moving paths of all MAM sensors as follows: $s_1 \to l_1 \to l_4$, $s_2 \to l_3 \to l_6$, and $s_3 \to l_5 \to l_2$. Compared with the D&C scheme in Section 3.2, we can reduce the overall energy consumption of MAM sensors from 20 to 18. In addition, our heuristic can balance the loads among MAM sensors since the standard deviation of their energy consumption is reduced from 3.3 (by the D&C scheme) to 0 in the example. This shows the effectiveness of our heuristic.

We finally discuss the rationale in phase 2. Phase 2 is invoked when $\mathcal{L}_u \neq \emptyset$ and by Corollary 1, only the MAM sensors in $\mathcal{M}$ are allowed to handle event locations in $\mathcal{L}_u$. This means that some of them have to visit more than one event location, where this problem is similar to TSP. Since a number of TSP heuristics [47] construct a spanning tree containing nodes to be visited, we are motivated by the same idea to make MAM sensors organize their spanning trees to select event locations in $\mathcal{L}_u$. In this way, if an MAM sensor can organize a "good" spanning tree, it can use such a TSP heuristic to calculate a shorter moving path to complete its dispatch schedule. Then, the question is how to organize a good spanning tree? There are two considerations in our design:

5. Notice that the moving path of an MAM sensor may not necessary be a Hamiltonian path. In other words, an MAM sensor can pass the same event location more than once. However, since TSP is widely used to address how to reduce the path length to visit a given set of locations, we also adopt a TSP heuristic in the paper.

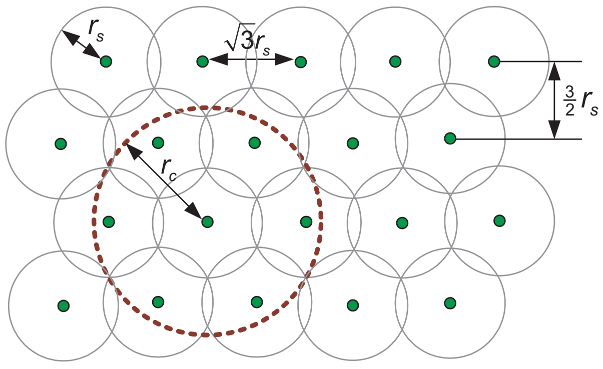


Fig. 4: The deployment of static sensors in the simulator.

1) The tree weights of all spanning trees should be similar so that the loads among MAM sensors can be balanced. In this case, the system lifetime can be extended [41].
2) The moving path of an MAM sensor could be proportional to the tree weight of its spanning tree, so we have to reduce the increase in the tree weight whenever adding an event location to that tree.

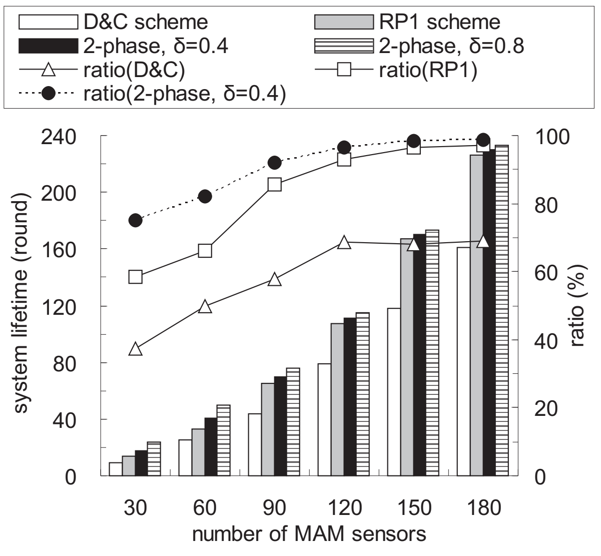
To address both considerations, in step 3 of the spanning-tree construction algorithm, we sort all trees in $\mathcal{T}$ and pick the first $\lceil \delta \cdot |\mathcal{T}| \rceil$ ones as candidates. In this way, when an event location, say, l_j joins any of these candidates, we can keep all tree weights in $\mathcal{T}$ as balanced as possible. Then, to satisfy the second consideration, l_j should join the tree, say, t_i such that the increase in t_i's tree weight is minimum. This can help reduce the moving path of t_i's root MAM sensor.

5 Experimental Results

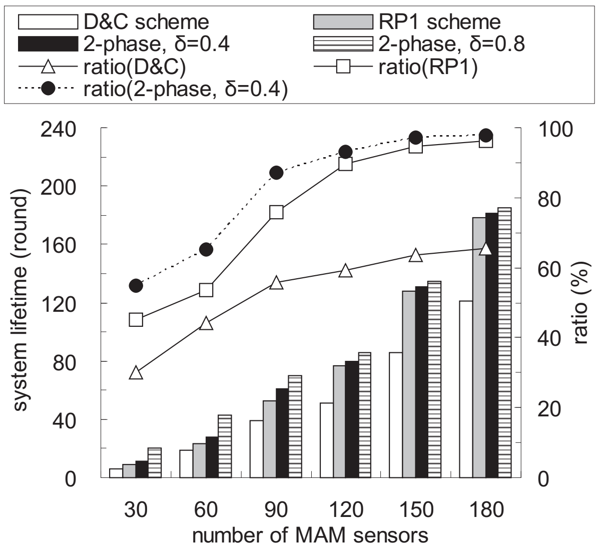
We develop a simulator by using C++ to measure the performances of different dispatch schemes. In the simulator, the sensing field is a 550 meters $\times$ 450 meters rectangle, on which 1000 static sensors are deployed by using a hexagon-like pattern, as shown in Fig. 4. The distance between any two adjacent static sensors is $\sqrt{3}r_s$, where r_s is the sensing range of a static sensor. To maintain network connectivity, we ensure that $r_c \geq \sqrt{3}r_s$, where r_c is the communication range of a static sensor. Thus, we set $r_s = 10$ meters and $r_c = 18$ meters. Given $\mathcal{A} = \{a_1, a_2, a_3\}$, a number of static sensors are arbitrarily selected as event locations (i.e., $\mathcal{L}$) in every round. We divide $\mathcal{L}$ into three groups $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ associated with attributes a_1, a_2, and a_3, respectively. Two scenarios are considered:

- **Equal scenario:** $|\mathcal{L}_1| = |\mathcal{L}_2| = |\mathcal{L}_3| = \frac{1}{3}|\mathcal{L}|$.
- **Biased scenario:** $|\mathcal{L}_1| = |\mathcal{L}_2| = \frac{1}{4}|\mathcal{L}|$ and $|\mathcal{L}_3| = \frac{1}{2}|\mathcal{L}|$.

There are also a number of MAM sensors (i.e., $\mathcal{S}$) arbitrarily deployed in the sensing field. Each MAM sensor is equipped with two batteries for its moving energy, where each battery has energy capacity of 1350 mAh (milliampere-hour). Thus, we have $e_i^{\text{init}} = 29160$ J (joule) for $i = 1..m$. The moving energy cost is 8.27 J per meter [48]. Since we aim at measuring the performances of different dispatch schemes, we do not estimate the energy spent on other operations such as sensing, communication, and analysis. In addition, the communication impairment (such as message loss or packet collision) is not considered in the experiments. We divide $\mathcal{S}$ into three groups $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ associated with attributes $\{a_1, a_2\}$, $\{a_2, a_3\}$, and $\{a_1, a_3\}$, respectively. Therefore, no single MAM sensor can handle every event location in $\mathcal{L}$.



(a) the equal scenario



(b) the biased scenario

Fig. 5: Comparison on the average system lifetime.

We compare our two-phase heuristic with the D&C scheme mentioned in Section 3.2. In addition, a dispatch scheme called *repeating phase 1 (RP1)* is also proposed for comparison. As its name suggested, the RP1 scheme iteratively repeats phase 1 in our heuristic to assign event locations to MAM sensors, until $\mathcal{L}$ becomes empty. We set the δ value to 0.4 and 0.8 in phase 2 to observe its effect on our heuristic.

5.1 Average System Lifetime

We first measure the average system lifetimes by different schemes. In each round, 150 static sensors are arbitrarily selected as $\mathcal{L}$. The number of MAM sensors ranges between 30 and 180. Each MAM sensor moves to the event locations assigned by the dispatch scheme and stays at its last-visiting location to wait for the next schedule. In Fig. 5, we use the result of our heuristic (with $\delta = 0.8$) as the comparison basis for all ratios.

Fig. 5(a) shows the average lifetimes in the equal scenario, where $|\mathcal{L}_1| = |\mathcal{L}_2| = |\mathcal{L}_3| = 50$. In general, the lifetime extends when the number of MAM sensors increases. The D&C scheme has the shortest lifetime because it handles different attributes of event locations separately, which narrows the view of each MAM sensor when deciding its destinations. In this case, not only the moving distances of some MAM sensors become longer but also the loads among all MAM sensors may be unbalanced. The RP1 scheme, on the other hand, allows MAM sensors to have a global view of event locations, and thus extends the lifetime compared with the D&C scheme. However, similar to the D&C scheme, the RP1 scheme also adopts an iterative concept to assign event locations to MAM sensors. Thus, it does not reduce the distance between any two event locations assigned to the same MAM sensor in different iterations. In this case, some of its assigned event locations could be far away from each other. This situation becomes worse when the number of MAM sensors is smaller than 150 (i.e., $|\mathcal{S}| < |\mathcal{L}|$).

Our heuristic always has the longest lifetime in Fig. 5(a). A larger δ value can help extend the lifetime since each MAM sensor can select a better tree (that is, a shorter distance to that tree) from more candidates. From the ratios in Fig. 5(a), our heuristic performs much better than the RP1 scheme when $|\mathcal{S}| \leq 120$. The reason is that more MAM sensors participate in phase 2 and the spanning-tree construction algorithm help them visit closer event locations. On the average, the lifetimes of the D&C scheme, the RP1 scheme, and our heuristic with $\delta = 0.4$ achieve only ratios of 58.6%, 82.7%, 90.4% of that of our heuristic with $\delta = 0.8$, respectively. The result verifies the effectiveness of our heuristic, especially with a larger δ value.

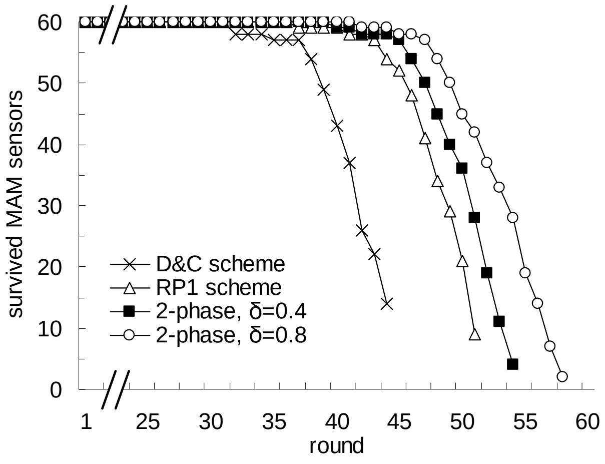
Fig. 5(b) shows the average lifetimes in the biased scenario, where $|\mathcal{L}_1| = |\mathcal{L}_2| = 38$ and $|\mathcal{L}_3| = 74$. In this scenario, since there are more event locations with attribute a_3, the energy of MAM sensors in both $\mathcal{S}_2$ and $\mathcal{S}_3$ will exhaust quickly[6]. The result in Fig. 5(b) is similar to that in Fig. 5(a), where the D&C scheme $<$ the RP1 scheme $<$ our heuristic with $\delta = 0.4 <$ our heuristic with $\delta = 0.8$. On the average, the lifetimes of the D&C scheme, the RP1 scheme, and our heuristic with $\delta = 0.4$ achieve only ratios of 53.1%, 75.8%, 82.5% of that of our heuristic with $\delta = 0.8$, respectively. The result indicates that our heuristic works more efficiently in the biased scenario. The reason is that phase 2 helps MAM sensors to select nearby event locations to visit, and thus reduces the energy costs of MAM sensors in $\mathcal{S}_2$ and $\mathcal{S}_3$ when visiting more event locations with attribute a_3.

5.2 Survived MAM Sensors

To observe how different dispatch schemes burden MAM sensors with different loads, we then estimate the number of survived MAM sensors in each round. In the experiment, we set $|\mathcal{S}| = 60$ and $|\mathcal{L}| = 90$, so some MAM sensors have to visit more than one event location in a round. We omit rounds 6 to 24 in both Fig. 6(a) and Fig. 7(a) since all MAM sensors are alive in these rounds.

Fig. 6 gives the number of survived MAM sensors in the equal scenario, where $|\mathcal{L}_1| = |\mathcal{L}_2| = |\mathcal{L}_3| = 30$. Since the D&C scheme burdens certain MAM sensors with heavier loads (as they are closer to some event locations), the first MAM sensor dies very early in the 32nd round. Then, after 44 rounds, the system lifetime terminates but there still remain 14 MAM sensors (without correct attributes to handle any event location). By iteratively finding a maximum Pareto-optimal matching to dispatch MAM sensors, the RP1 scheme postpones the dead time of the first MAM sensor to the 37th round and leaves 9 MAM sensors in the last round. Our heuristic performs the best in the equal scenario, where a larger δ value helps extend the lifetime. Specifically, our heuristic not only drains the first MAM sensor of its energy quite late (after 40 rounds) but

6. Since the MAM sensors in $\mathcal{S}_1$ do not have attribute a_3, they cannot help handle these event locations and thus burden other MAM sensors with heavier loads.

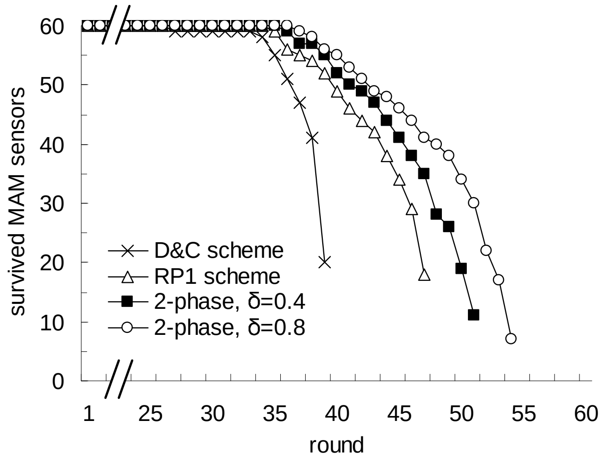


(a) survived MAM sensors (rounds 6–24 are omitted)

dispatch scheme	1st MAM sensor dies (round #)	lifetime (round)	final survived MAM sensors
D&C scheme	32nd	44	14
RP1 scheme	37th	51	9
2-phase, $\delta = 0.4$	40th	54	4
2-phase, $\delta = 0.8$	42nd	58	2

(b) statistics

Fig. 6: Comparison on the number of survived MAM sensors in the equal scenario.



(a) survived MAM sensors (rounds 6–24 are omitted)

dispatch scheme	1st MAM sensor dies (round #)	lifetime (round)	final survived MAM sensors
D&C scheme	27th	39	20
RP1 scheme	35th	47	18
2-phase, $\delta = 0.4$	36th	51	11
2-phase, $\delta = 0.8$	37th	54	7

(b) statistics

Fig. 7: Comparison on the number of survived MAM sensors in the biased scenario.

also leaves fewer MAM sensors (no more than 4) in the last round. This result shows that our heuristic can better use MAM sensors to analyze events, compared with other schemes.

Fig. 7 gives the number of survived MAM sensors in the biased scenario, where $|\mathcal{L}_1| = |\mathcal{L}_2| = 23$ and $|\mathcal{L}_3| = 44$. The result is similar to Fig. 6. However, since $\mathcal{L}_3$ contains more event locations, which burdens the MAM sensors in both $\mathcal{S}_2$ and $\mathcal{S}_3$ with heavier loads, the first MAM sensor dies earlier in all schemes. In addition, by comparing Fig. 7(b) with Fig. 6(b), there are more final survivals in the biased scenario. All of these survivals must belong to $\mathcal{S}_1$ since they cannot handle event locations with attribute a_3. In sum, our heuristic performs the best in the biased scenario, and a larger δ value helps improve the performance.

5.3 Moving Energy Consumption

Finally, we evaluate the energy consumption of MAM sensors, where $|\mathcal{S}| = 90$ and $|\mathcal{L}|$ ranges between 30 and 150. Both the average and standard deviation of energy consumption of MAM sensors are measured in the experiment. For all ratios in Figs. 8 and 9, we use the result of the D&C scheme as the comparison basis.

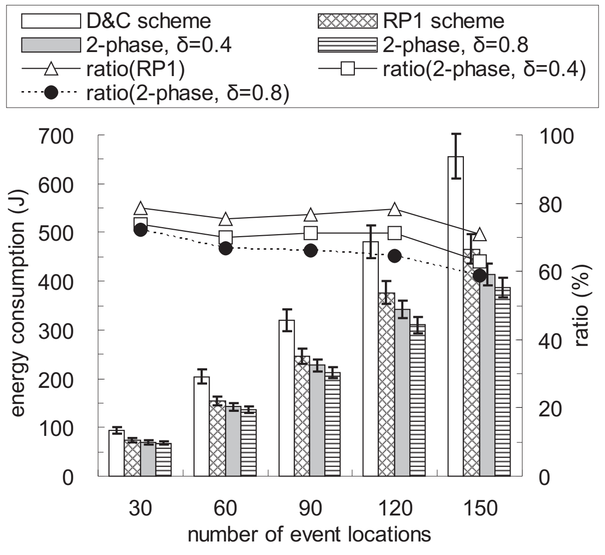
Fig. 8 shows the energy consumption of MAM sensors in the equal scenario. As mentioned earlier, the D&C scheme narrows the view of each MAM sensor when deciding its destinations. Thus, this scheme always makes MAM sensors spend the most energy and it has the largest standard deviation (in other words, MAM sensors are burdened with very unbalanced loads). With the help of phase 2, our heuristic not only saves the moving energy of MAM sensors but also significantly reduces the standard deviation, as comparing with the RP1 scheme. This result shows the effectiveness of phase 2.

From Fig. 8, δ has different effects on the average and standard deviation of energy consumption in our heuristic. In particular, a larger δ value reduces the energy cost since it allows each MAM sensor to have more candidate spanning trees to select in phase 2. However, the risk that trees become unbalanced could also arise, whose effect is shown in Fig. 8(b). On the average, the RP1 scheme, our heuristic with $\delta = 0.4$, and our heuristic with $\delta = 0.8$ can save 24.0%, 30.2%, and 34.3% of energy cost comparing with the D&C scheme, respectively. Meanwhile, the RP1 scheme, our heuristic with $\delta = 0.4$, and our heuristic with $\delta = 0.8$ can reduce 39.4%, 73.1%, and 68.7% of the standard deviation of energy consumption comparing with the D&C scheme, respectively. This result demonstrates the effectiveness of our heuristic in the equal scenario.

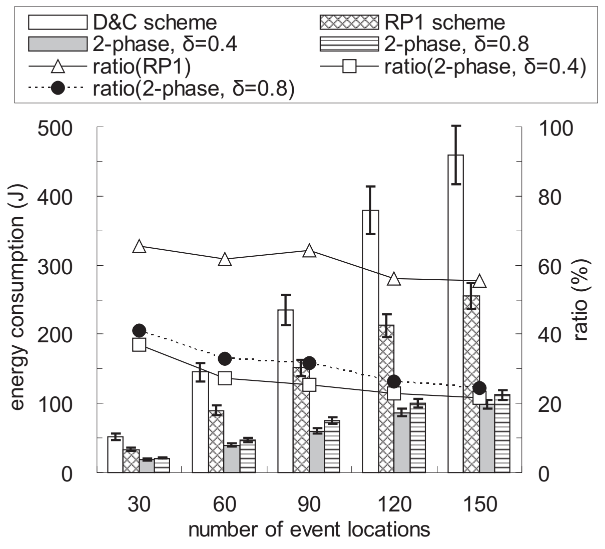
Fig. 9 shows the energy consumption of MAM sensors in the biased scenario, where two interesting phenomena arise:

- By comparing Fig. 9(a) with Fig. 8(a), the energy cost of the D&C scheme even reduces when $|\mathcal{L}| = 150$ in the biased scenario[7]. The reason is that the D&C scheme burdens some MAM sensors (especially those in $\mathcal{S}_2$ and $\mathcal{S}_3$) with heavier loads, leading other MAM sensors to spend less energy. The above argument can be verified by both Fig. 7 and Fig. 9(b), where a lot of MAM sensors survive in the last round and the standard deviation of energy consumption among all MAM sensors is much larger than other schemes.
- By comparing Fig. 9(b) with Fig. 8(b), the ratios become larger in the biased scenario when $|\mathcal{L}| \leq 90$. In this case, since a half of event locations have attribute a_3, the D&C scheme could calculate a more "balanced" matching for the MAM sensors in both $\mathcal{S}_2$ and $\mathcal{S}_3$ to handle these event locations. Because the comparison basis (i.e., the standard deviation of the D&C scheme) reduces, all ratios increase accordingly when $|\mathcal{L}| \leq 90$.

7. In other three schemes, the energy costs increase in the biased scenario.



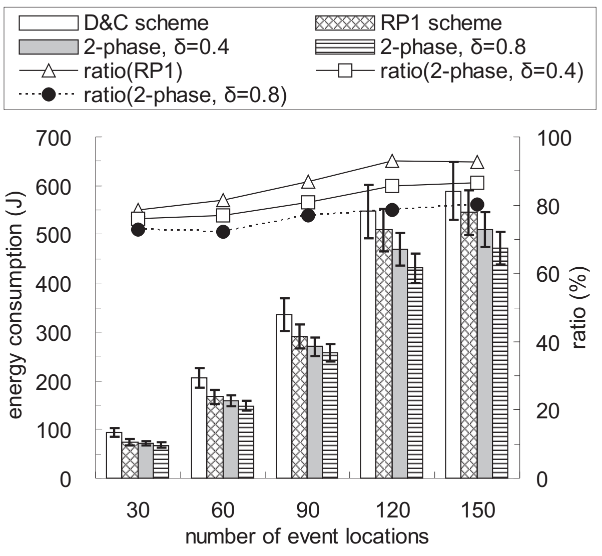
(a) average of energy consumption



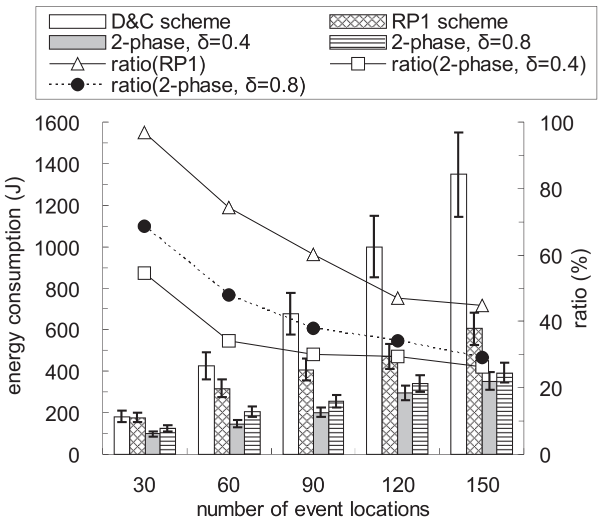
(b) standard deviation of energy consumption

Fig. 8: Comparison on the energy consumption of MAM sensors in the equal scenario.

On the average, the RP1 scheme, our heuristic with $\delta = 0.4$, and our heuristic with $\delta = 0.8$ can save 13.5%, 18.8%, and 23.8% of energy cost comparing with the D&C scheme, respectively. Meanwhile, the RP1 scheme, our heuristic with $\delta = 0.4$, and our heuristic with $\delta = 0.8$ can reduce 35.3%, 65.1%, and 56.4% of the standard deviation of energy consumption comparing with the D&C scheme, respectively. This result verifies that our heuristic still performs well in the biased scenario.



(a) average of energy consumption



(b) standard deviation of energy consumption

Fig. 9: Comparison on the energy consumption of MAM sensors in the biased scenario.

6 Conclusions and Future Work

This paper has formulated the MAM sensor dispatch problem in a hybrid WSN. To solve this NP-complete problem, a two-phase heuristic is developed by reducing and balancing the energy consumption of MAM sensors. In phase 1, we assign event locations to MAM sensors in a one-to-one manner by finding a maximum Pareto-optimal matching in a weighted bipartite graph. Since there could remain unassigned event locations, a spanning-tree construction algorithm is proposed in phase 2 to handle this case. Experimental results show that our heuristic can extend the system lifetime compared with other schemes.

Below, we present some future research issues. First, we aim at path efficiency of MAM sensors in our heuristic, so they are scheduled to reduce and balance the moving distances. However, since analyzing different attributes of events may require different amounts of time, it deserves further investigation to schedule MAM sensors such that the overall time spent to complete the assignment is also minimum. In this case, we should consider not only the moving costs of MAM sensors but also the time to analyze events. Second, a distributed version of our heuristic can be designed to let MAM sensors work in a decentralized fashion. Finally, a new challenge arises when events also have multiple attributes.

References

[1] Y.C. Wang, F.J. Wu, and Y.C. Tseng, “Mobility management algorithms and applications for mobile sensor networks,” *Wireless Comm. and Mobile Computing*, vol. 12, no. 1, pp. 7–21, 2012.
[2] G. Song, Z. Wei, W. Zhang, and A. Song, “A hybrid sensor network system for home monitoring applications,” *IEEE Trans. Consumer Electronics*, vol. 53, no. 4, pp. 1434–1439, 2007.
[3] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, “iMouse: an integrated mobile surveillance and wireless sensor system,” *Computer*, vol. 40, no. 6, pp. 60–66, 2007.
[4] L. Cheng, C.D. Wu, and Y.Z. Zhang, “Indoor robot localization based on wireless sensor networks,” *IEEE Trans. Consumer Electronics*, vol. 57, no. 3, pp. 1099–1104, 2011.
[5] N. Bartolini, T. Calamoneri, T.F. La Porta, and S. Silvestri, “Autonomous deployment of heterogeneous mobile sensors,” *IEEE Trans. Mobile Computing*, vol. 10, no. 6, pp. 753–766, 2011.
[6] D.P. Eickstedt, M.R. Benjamin, H. Schmidt, and J.J. Leonard, “Adaptive control of heterogeneous marine sensor platforms in an autonomous sensor network,” *Proc. IEEE/RSJ Int’l Conf. Intelligent Robots and Systems*, pp. 5514–5521, 2006.
[7] S. Ferrari, R. Fierro, and D. Tolic, “A geometric optimization approach to tracking maneuvering targets using a heterogeneous mobile sensor network,” *Proc. IEEE Conf. Decision and Control*, pp. 1080–1087, 2009.
[8] M. Hofmeister, M. Kronfeld, and A. Zell, “Cooperative visual mapping in a heterogeneous team of mobile robots,” *Proc. IEEE Int’l Conf. Robotics and Automation*, pp. 1491–1496, 2011.
[9] R. Rao and G. Kesidis, “Purposeful mobility for relaying and surveillance in mobile ad hoc sensor networks,” *IEEE Trans. Mobile Computing*, vol. 3, no. 3, pp. 225–231, 2004.
[10] Y.C. Wang, W.C. Peng, M.H. Chang, and Y.C. Tseng, “Exploring load-balance to dispatch mobile sensors in wireless sensor networks,” *Proc. IEEE Int’l Conf. Computer Comm. and Networks*, pp. 669–674, 2007.
[11] D.J. Abraham, K. Cechlarova, D.F. Manlove, and K. Mehlhorn, “Pareto optimality in house allocation problems,” *Proc. Int’l Conf. Algorithms and Computation*, pp. 1163–1175, 2005.
[12] Z.J. Haas and B. Liang, “Ad hoc mobility management with uniform quorum systems,” *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 228–240, 1999.
[13] N. Li, J.C. Hou, and L. Sha, “Design and analysis of an MST-based topology control algorithm,” *IEEE Trans. Wireless Comm.*, vol. 4, no. 3, pp. 1195–1206, 2005.
[14] H. Nishiyama, T. Ngo, N. Ansari, and N. Kato, “On minimizing the impact of mobility on topology control in mobile ad hoc networks,” *IEEE Trans. Wireless Comm.*, vol. 11, no. 3, pp. 1158–1166, 2012.
[15] T. Camp, J. Boleng, and V. Davies, “A survey of mobility models for ad hoc network research,” *Wireless Comm. and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.
[16] Q. Li and D. Rus, “Communication in disconnected ad hoc networks using message relay,” *J. Parallel Distributed Computing*, vol. 63, pp. 75–86, 2003.
[17] D. Niyato and P. Wang, “Optimization of the mobile router and traffic sources in vehicular delay-tolerant network,” *IEEE Trans. Vehicular Technology*, vol. 58, no. 9, pp. 5095–5104, 2009.
[18] H. Dang and H. Wu, “Clustering and cluster-based routing protocol for delay-tolerant mobile networks,” *IEEE Trans. Wireless Comm.*, vol. 9, no. 6, pp. 1874–1881, 2010.
[19] L. Vig and J.A. Adams, “Multi-robot coalition formation,” *IEEE Trans. Robotics*, vol. 22, no. 4, pp. 637–649, 2006.
[20] K.H. Park, Y.J. Kim, and J.H. Kim, “Modular Q-learning based multi-agent cooperation for robot soccer,” *Robotics and Autonomous Systems*, vol. 35, no. 2, pp. 109–122, 2001.
[21] K.S. Hwang, Y.J. Chen, and C.H. Lee, “Reinforcement learning in strategy selection for a coordinated multirobot system,” *IEEE Trans. Systems, Man and Cybernetics — Part A: Systems and Humans*, vol. 37, no. 6, pp. 1151–1157, 2007.
[22] J. Mugan and B. Kuipers, “Autonomous learning of high-level states and actions in continuous environments,” *IEEE Trans. Autonomous Mental Development*, vol. 4, no. 1, pp. 70–86, 2012.
[23] M.G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A.J. Kleywegt, “Simple auctions with performance guarantees for multi-robot task allocation,” *Proc. IEEE/RSJ Int’l Conf. Intelligent Robots and Systems*, pp. 698–705, 2004.

[24] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: a survey and analysis," *Proc. of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[25] B.P. Gerkey and M.J. Mataric, "Sold!: auction methods for multirobot coordination," *IEEE Trans. Robotics and Automation*, vol. 18, no. 5, pp. 758–768, 2002.

[26] L. Luo, N. Chakraborty, and K. Sycara, "Competitive analysis of repeated greedy auction algorithm for online multi-robot task assignment," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 4792–4799, 2012.

[27] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, pp. 1957–1962, 2003.

[28] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 1145–1150, 2005.

[29] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 34–42, 2003.

[30] G. Wang, G. Cao, and T.F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.

[31] G. Tan, S.A. Jarvis, and A.M. Kermarrec, "Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks," *IEEE Trans. Mobile Computing*, vol. 8, no. 6, pp. 836–848, 2009.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] G. Wang, G. Cao, T.F. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," *Proc. IEEE INFOCOM*, pp. 2302–2312, 2005.

[39] 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.

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

[41] 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.

[42] S. Zhang, J. Cao, L. Chen, and D. Chen, "Accurate and energy-efficient range-free localization for mobile sensor networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 6, pp. 897–910, 2010.

[43] S.Q. Zheng, J.S. Lim, and S.S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 103–110, 1996.

[44] M. Udi, *Introduction to Algorithms: A Creative Approach*, Addison-Wesley Publishing Company, 1989.

[45] H.W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[46] J.E. Hopcroft and R.M. Karp, "A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Computing*, vol. 2, pp. 225–231, 1973.

[47] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics, 2007.

[48] M. Rahimi, H. Shah, G.S. Sukhatme, J. Heideman, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," *Proc. IEEE Int'l Conf. Robotics and Automation*, pp. 19–24, 2003.