An Energy-Balanced Heuristic for Mobile Sink Scheduling in Hybrid WSNs

ZhangBing Zhou, *Member, IEEE*, Chu Du, Lei Shu, *Member, IEEE*, Gerhard Hancke, *Senior Member, IEEE*, Jianwei Niu, *Senior Member, IEEE*, and Huansheng Ning, *Senior Member, IEEE*

***Abstract*—Wireless sensor networks (WSNs) are integrated as a pillar of collaborative Internet of Things (IoT) technologies for the creation of pervasive smart environments. Generally, IoT end nodes (or WSN sensors) can be mobile or static. In this kind of hybrid WSNs, mobile sinks move to predetermined sink locations to gather data sensed by static sensors. Scheduling mobile sinks energy-efficiently while prolonging the network lifetime is a challenge. To remedy this issue, we propose a three-phase energy-balanced heuristic. Specifically, the network region is first divided into grid cells with the same geo-graphical size. These grid cells are assigned to clusters through an algorithm inspired by the k-*d*imensional tree algorithm, such that the energy consumption of each clus-ter is similar when gathering data. These clusters are adjusted by (de)allocating grid cells contained in these clusters, while considering the energy consumption of sink movement. Consequently, the energy to be consumed in each cluster is approximately balanced considering the energy consumption of both data gathering and sink movement. Experimental evaluation shows that this technique can generate an optimal grid cell division within a limited time of iterations and prolong the network lifetime.**

***Index Terms*—Energy-balanced heuristics, grid cell, hybrid wireless sensor networks (WSNs), mobile sinks.**

This work was supported in part by the National Natural Science Foundation of China under Grant 61379126, Grant 61401107, Grant 61572060, and Grant 61170296; in part by the Scientific Research Foundation for Returned Scholars, Ministry of Education of China; and in part by the Fundamental Research Funds for the Central Universities. Paper no. TII-15-0703.
(Corresponding author: ZhangBing Zhou.)

Z. Zhou is with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China, also with the School of Information Engineering, China University of Geosciences, Beijing 100083, China, and also with the Department of Computer Science, TELECOM SudParis, Evry 91011, France (e-mail: zhangbing.zhou@gmail.com).

C. Du is with the 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang 050081, China (e-mail: duchusac@gmail.com).

L. Shu is with Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China (e-mail: lei.shu@ieee.org).

G. Hancke is with the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria, Hatfield 0028, South Africa (e-mail: gerhard.hancke@up.ac.za).

J. Niu is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: niujianwei@buaa.edu.cn).

H. Ning is with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: ninghuansheng@ustb.edu.cn).

I. INTRODUCTION

ALONG with the advent and rapid development of the Internet of Things (IoT), it is envisioned that by the year 2020, there will be over 50 billion connected smart things in the world, and these smart things could act as sensors to support widespread domain applications. The interconnection of smart things will provide real-world sensory data for driving higher efficiencies and streamlining business practices. In fact, the Internet nowadays is becoming the platform of communication, computation, coordination, and cooperation for machines and smart objects [1]. The achievement of the IoT vision leverages smart things (such as sensors and actuators) to detect environmental variables and to response for supporting domain applications. Smart things communicate primarily in a wireless manner and construct a sensor network [2]. In fact, wireless sensor networks (WSNs) have become an important research topic in the recent decades and have been applied in many application scenarios like target tracking [3]. As argued in [4], WSNs are a key component at the perception layer of the IoT architecture, and serve as the data acquisition interface for IoT with the physical environment. Generally, WSNs are well recognized as an important enabling technique for achieving the vision of IoT [1].

Empowered by embedded computing and wireless communication techniques, sensors can move around when installed on mobile equipment. Typically, mobile sensors are resource-rich devices with more energy, higher communication power, and more powerful sensing and computational capabilities. However, mobile sensors still have energy, computational storage, and other constraints [5]. In this hybrid sensor network environment, static sensors are responsible for sensing environmental variables, while mobile sensors, also called IoT mobile sinks [6], move to designated locations for gathering data sensed by static sensors. The visiting frequency of mobile sinks depends on the data buffer on static sensors and data sensing rates. Generally, when a mobile sink moves to a (predetermined) sink location, it stops and broadcasts its arrival. Thereafter, static sensors start to transfer their sensory data to this mobile sink. Leveraging this data gathering (or collection) strategy, mobile sinks collaborate to fulfill domain applications, such as battle field surveillance and habitat monitoring [7]. Note that the data aggregation mechanisms of mobile sinks depend on specific requirements of domain applications.

Without loss of generality, the energy consumed for data gathering is assumed the same for all mobile sinks when the number of static sensors is the same. In this context, scheduling mobile sinks efficiently, while prolonging the network lifetime as much as possible, is one of the major research challenges. It is worth mentioning that the bulk of energy consumption for mobile sinks is to support their movement [16]. Energy harvesting techniques can be used to replenish the energy of mobile sinks, but the energy replenished is usually much less than that consumed by mobile sinks. Therefore, the lifetime of a hybrid WSN depends on that of mobile sinks. Generally, the lifetime of WSNs can be defined in different ways. This paper adopts the widely used one, which is the time when the first mobile sink depletes its energy [7]. With the mechanisms of duty-cycle and energy harvesting, static sensors usually can survive until the first mobile sink depletes its energy. In this paper, we study the scheduling mechanism of mobile sinks to cover the whole network region in an energy-efficient and -balanced manner.

The scheduling of mobile sinks is an important research topic in hybrid WSNs and techniques have been proposed to address this problem from different perspectives [6], [8]–[10]. An inspiring survey about the evolution of sink mobility issues in WSNs has been presented in [27]. It is argued that sink mobility is a recent trend to mitigate the network performance issue, which can hardly be solved properly by static WSNs. Generally, single (or multiple) sink(s) mobility scheduling is a problem of NP-hard. Consequently, heuristics are adopted mostly for the scheduling of sink movement. Intuitively, the task of mobile sink scheduling can be modeled as the well-known (multiple) traveling salesman problem [(m)TSP]. Since TSP (or mTSP) is NP-hard and can hardly be used when the number of static and mobile sinks is relatively large, heuristics are adopted mostly for scheduling routes (or paths) of feasible [11]. For instance in [8], mobile sensors are assumed to have multiple capabilities. They can move to appropriate event locations, which are unpredictable and detected by static sensors in real time, to conduct a more in-depth analysis. A two-phase heuristic is proposed for assigning mobile sensors to event locations, while extending the system lifetime. This method is interesting and inspiring for the technique developed in this paper. However, (mobile) sensors normally have similar set of capabilities in most cases, and how to efficiently assign mobile sensors to event locations, where multiple attributes need to be examined, is still open. In [9], it is envisioned that for a very large region to be monitored, it is almost impossible to cover the whole region using static sensors. In this setting, actuators are used to mitigate this issue for improving the area coverage, target detection, and many other tasks. Balancing the workload of actuators is important in this context. After the initial deployment, their workload is computed dynamically in light of the partition result of Voronoi diagrams. When an imbalance is detected, actuators are moved to achieve a balanced load distribution. Voronoi diagrams are irregular somehow, and hence, the partition may not be optimal when the path traversing cost is unnegligible. Besides, we assume in this research that the network region can be covered by static sensors. The technique in [6] proposes to schedule one single mobile sink to travel over sink sites (or positions) for gathering environmental variables with delay constraints. The mobile sink is assumed to have limitless energy compared with static sensors. The technique in [10] mitigates the region coverage issue when a region is relatively large and mobile sensors move independently following decentralized control principles. Generally, current techniques have studied the scheduling of mobile sinks in hybrid WSNs. These include the dispatch of mobile sinks with multicapabilities, in a sparse (or) static sensor deployment situation. However, how to schedule mobile sinks efficiently while prolonging the network lifetime is still a challenge to be explored further.

This research aims to propose an efficient technique for scheduling mobile sinks of a hybrid WSN in the framework of IoT, while balancing the workload of these IoT mobile sinks, such that the network lifetime is prolonged. The region to be monitored is assumed to be covered by static sensors and has no holes (or obstacles) [12]. Static sensors sense and buffer their newly generated data, and these data are assumed delay-tolerant to applications. In a nutshell, we follow a divide-and-conquer strategy to develop a technique, including the following steps.

1) *Region Division to Grid Cells:* The region to be monitored is divided into grid cells, whose size is proportional to the communication radius of the static sensors. Grid cells are the basic unit for mobile sinks to gather data sensed by static sensors. To minimize the energy consumption of mobile sink movement, there are some (possibly one) sink positions to be predetermined in each grid cell, such that a mobile sink is required to move to these sink positions only for gathering sensory data.
2) *Grid Division w.r.t. Data Gathering Energy Cost:* Grid cells are the same in size, but may differ in the number of static sensors contained (especially when static sensors are in a skewed distribution). This indicates that the energy consumption for mobile sinks of gathering data may differ considerably for different grid cells. In this step, grid cells are grouped into clusters using an algorithm inspired by the k-$d$imensional tree algorithm. The number of clusters is set to the same number as that of mobile sinks. Hence, the data gathering effort is almost the same (within an allowed threshold of difference) in each cluster.
3) *Cluster Adjustment w.r.t. Sink Movement Cost:* In the previous step, the energy consumption of sink movement has not been considered. In fact, the energy consumption of mobile sinks is different (to a large extent), especially when static sensors are not distributed evenly. By taking this into account, we propose to adjust the initial grid division by considering the impact of the movement cost of mobile sinks, such that the energy consumption of data gathering and sink movement is almost the same for all clusters. Consequently, the energy consumption is almost the same for each mobile sink, and the lifetime of each mobile sink does not differ to a certain extent. Therefore, the network lifetime is prolonged.

Note that when the grid cell division and adjustment procedures have been applied, no update needs to be enacted afterward. Compared with the state of the art, our technique ensures that the energy consumption of all mobile sinks is almost the similar, which causes the life of mobile sinks not to differ much. Hence, the lifetime of the network is prolonged.

This paper is organized as follows. Section II introduces the network model. Section III discusses related work. Section IV presents the network division strategy. Section V introduces the clustering technique considering the energy consumption of data collection only, while Section VI proposes to adjust clusters through transferring grid cells between clusters when considering the energy consumption of mobile sink movement. Section VII presents the experimental evaluation, and Section VIII makes a conclusion and presents future directions.

II. PRELIMINARY

IoT end nodes (or smart things) are primarily wirelessly connected and construct a network, while these nodes can be mobile or static [2]. In fact, IoT end nodes correspond to the sensors in WSNs. As mentioned above, a hybrid WSN contains two types of sensors: 1) static sensors; and 2) mobile sinks. Static sensors are deployed in the network region for sensing environmental variables, while mobile sinks move, following predetermined paths, to designated locations for gathering data sensed by static sensors. Sensors are aware of their locations, which can be achieved through global positioning systems or other localization techniques. Note that the location of sensors is obtained during the network initialization phase, and does not need to be updated afterward. Hence, the energy consumption of location acquisition is relatively small and negligible. Static sensors may be deployed unevenly in the network region. Without loss of generality, we assume that a hybrid WSN has no holes, and static sensors (or mobile sinks) are assumed to possess the same capabilities. Mobile sinks have much more capabilities, including energy and communication radius, than those of static sensors, although still have limitations in their capabilities. Since static sensors are responsible for sensing environmental variables and sending data to mobile sinks, while mobile sinks move around and gather sensory data by static sensors, the energy consumption of mobile sinks are much higher than that of static sensors. Specifically, a mobile sink moves to a grid cell, stops at prespecified sink positions, and gathers data sensed by static sensors. The energy consumption occurs during the sink movement and data gathering phases. Since the energy of static sensors can be replenished by the techniques of energy harvesting, the lifetime of the hybrid WSNs is determined by the lifetime of mobile sinks (rather than that of static sensors).

Generally, this work is to propose a mobile sink scheduling mechanism, which aims to prolong the lifetime of the hybrid WSNs as much as possible. The energy consumption of mobile sinks includes two parts: 1) sink movement; and 2) data gathering. Given a network, the energy consumption of data gathering is proportional to the number of static sensors contained in this network, while that of sink movement is proportional to the distance of sink movement. Prolonging the network lifetime requires that mobile sinks should move as short a distance as possible, and the energy consumption of mobile sinks should be as close as possible. To mitigate this problem, we divide the network region into grid cells, and group grid cells into clusters, such that each mobile sink has similar energy consumption when considering the cost of both data gathering and mobile sink movement. Below we list symbols and notations used in the subsequent sections.

S	a hybrid WSN with n static sensors and m mobile sinks. Note that n is much bigger than m in domain applications.
r	the data communication radius of static sensors. Note that the communication radius of mobile sinks is typically much bigger than that of static sensors.
g_i, gSide:	g_i is the grid cell i divided from the network S, and gSide is the side length of g_i. Without loss of generality, all grid cells are equivalent in their geographic size and a grid cell is a square in shape.
c_i:	the cluster i ($\mathrm{i} \in (0,\ \mathrm{m}]$). The cluster c_i is built through grouping the relevant grid cells.
$c_i \cdot$ gnum:	the number of grid cells contained in a certain cluster c_i.
GN(i,j):	the set of grid cells belonging to the cluster c_i and neighboring the cluster c_j($\mathrm{j} \in (0,\ \mathrm{m}]$). Note that c_j denotes an adjacent cluster to the cluster c_i.
gn$_k(i,j)$:	the grid cell k in GN(i,j), which is numbered from top to bottom, and from left to right.
$g_i \cdot$ ECD:	the energy consumption of a mobile sink when gathering data sensed by static sensors in the grid cell g_i. $g_i \cdot$ ECD is proportional to the number of static sensors in g_i, since sensory data of all sensor nodes contained in g_i are gathered. Note that there may be a minor difference in energy consumption due to the difference in distance between static sensors and the corresponding mobile sink. For simplicity, this is not considered in this technique.
$c_i \cdot$ ECM:	the energy consumption of a mobile sink when moving within a cluster c_i for traversing all grid cells contained in c_i. Generally, $c_i \cdot$ ECM is proportional to the length of the path that a mobile sink traverses in c_i.
uECM:	the energy consumption of a mobile sink when moving from a grid cell g_i to its neighboring grid cell g_j. It is worth mentioning that the movement direction can only be horizontal or vertical.
$c_i \cdot$ EC:	the energy that a mobile sink consumes when it traverses all grid cells in the cluster c_i for gathering data sensed by static sensors. $c_i \cdot$ EC is composed of $c_i \cdot$ ECM and all $g_i \cdot$ ECD, where the grid cell g_i belongs to the cluster c_i. Specifically, $g_i \cdot$ ECD represents the energy consumption of gathering all sensory data in the cluster c_i, while $c_i \cdot$ ECM represents that of sink movement within c_i.

As mentioned above, this technique is to prolong the network lifetime, which requires the lifetime of mobile sinks to be as close as possible. This mandates that the difference of $c \cdot$ EC for each mobile sink should be as small as possible. We use the formula in the following to reflect this requirement, where V represents the variance of $c \cdot$ EC. Generally, a smaller value of V means that the difference between $c \cdot$ EC for mobile sinks is relatively smaller. Specifically, $(c \cdot \mathrm{EC} - \frac{\sum_{i=1}^{m} c_i \cdot \mathrm{EC}}{m})^2$ is smaller only when the bias of $c \cdot$ EC for all mobile sinks is

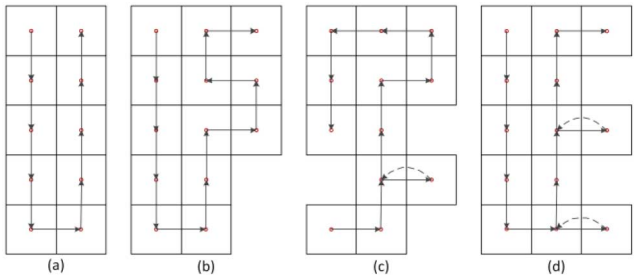

Fig. 1. Diversification of $c_i \cdot$ ECM when adjusting grid cells.

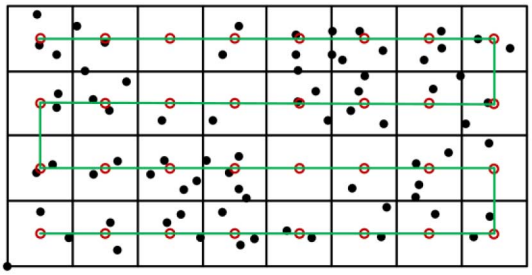

Fig. 2. Grid cell division with a small side length (i.e., $\sqrt{2}r/2$), where the black and solid points represent static sensors, while the red and circled points represent sink locations. The points in Fig. 3 are the same as these in Fig. 2.

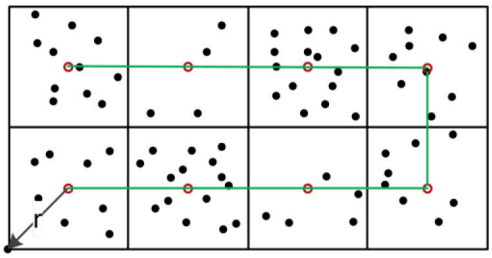

Fig. 3. Grid cell division where the side length is set to $\sqrt{2}r$.

smaller. This reflects that the energy consumption of all mobile sinks does not differ to a certain extent. Therefore, the minimization of V is used as the criterion, when grouping grid cells into clusters and assigning clusters to mobile sinks for gathering data of static sensors

$$V = \frac{\sum_{i=1}^{m} \left(c_i \cdot \text{EC} - \frac{\sum_{i=1}^{m} c_i \cdot \text{EC}}{m} \right)^2}{m}. \quad (1)$$

Intuitively, if grid cells are smaller (gSide is set to a relatively smaller value), $c \cdot$ EC should be smaller as well. In this setting, it may be possible to make $c \cdot$ EC similar in value for all clusters through the mechanisms proposed in Sections IV and V. On the other hand, as illustrated in Figs. 2 and 3, when grid cells are too small, certain energy consumed for the sink movement may be unnecessary. This means that more energy has to be consumed and the network lifetime may be impacted negatively. Hence, grid cells should be set to the most appropriate size (i.e., gSide $= \sqrt{2}r$), to prolong the network lifetime, although V may not be the minimum in this setting.

Before presenting our technique, we would like to discuss more about $c_i \cdot$ ECM. Since no holes exist in the network region, given a cluster c_i, a mobile sink needs to move $c_i \cdot$ gnum $- 1$ steps to traverse all grid cells contained in most cases, as shown in Fig. 1(a) and (b). However, for some other cases as illustrated in Fig. 1(c) and (d), the mobile sink is mandated to move $c_i \cdot$ gnum or $c_i \cdot$ gnum $+ 1$ steps for a traverse. It is worth noting that the cases of Fig. 1(c) and (d) do not exist after initial clustering of grid cells (see Section V). Furthermore, we have tried to avoid the cases as shown in Fig. 1(c) and (d) during the reallocation of grid cells between clusters (see Section VI). This means that the cases of Fig. 1(c) and (d) are rare in final clusters. For simplicity, we set $c_i \cdot$ ECM $= (c_i \cdot \text{gnum} - 1) \times$ uECM in the following sections.

Table I
$c_i \cdot$ ECD AND $c_j \cdot$ ECM ADJUSTMENT WHEN REALLOCATING GRID CELLS $\text{gn}_1(i,j), \ldots, \text{gn}_k(i,j)$ FROM THE CLUSTER c_i TO A NEIGHBORING CLUSTER c_j

	Adjustment		Adjustment
c_i.ECD	$-\sum_{h=1}^{k} (\text{gn}_h(i,j))$	c_j.ECD	$+\sum_{h=1}^{k} (\text{gn}_h(i,j))$
c_i.ECM	$-k \times u$ECM	c_j.ECM	$+k \times u$ECM

When reassigning grid cells $\text{gn}_1(i,j), \ldots, \text{gn}_k(i,j)$ from a cluster c_i to a neighboring cluster c_j, $c_i \cdot$ ECD is updated depending on k and $\sum_{i=1}^{k} \text{gn}_i(i,j) \cdot$ ECD. The formulae for updating c_j, $c_{\ i} \cdot$ ECD and $c_j \cdot$ ECM for this reallocation are presented in Table I, where the first row represents the energy consumption of data gathering, while the second row represents that of sink movement, for the transformation from the cluster c_i to its neighboring cluster c_j.

III. RELATED WORK AND COMPARISON

In [8], a hybrid WSN is composed of static and mobile sensors. Each static sensor is assumed to detect one kind of attribute, while each mobile sensor can analyze multiple (not all normally) attributes of events. Mobile sensors are required moving to appropriate event locations, which are detected by static sensors, for conducting more in-depth analysis. Events are assumed to be happening independently and unpredictably. To mitigate this multiattribute mobile (MAM) sensor dispatch problem, a two-phase heuristic is proposed for the assignment of mobile sensors to event locations while extending the system lifetime. Specifically, MAM sensors are assigned to event locations in a one-to-one fashion initially, and a spanning-tree construction algorithm is adopted for dispatching MAM sensors to unassigned event locations afterward. This research is promising and has inspired the development of our technique. However, (mobile) sensors normally have a similar set of capabilities, and a method to efficiently assign MAM sensors to event locations, where multiple attributes are to be examined, is not explored. The authors have studied the mobility management algorithms of mobile sensors and reviewed some existing WSN platforms in [15]. Besides, the authors have studied the mobile sink scheduling problem in centralized and distributed situations [16]. In centralized environments, static sensors are divided into clusters, and the traverse of mobile sensors is constructed using the minimum spanning tree. In distributed counterparts, static sensors are divided according

to a grid structure, which is similar to our grid-division strategy presented in Section IV. The difference between this work and our technique is that the dispatching of mobile sensors in this work is triggered in an event-oriented manner, where a part of the network region is normally to be investigated. By contrast, the coverage of the whole network region is our concern.

In [9], the authors argue that in certain applications like space exploration, static sensors can hardly cover the entire target region and cannot ensure the network connectivity. In this context, mobile actuators are used for mitigating this network architecture challenge. Balancing the workload of actuators is a critical issue for prolonging the network lifetime. To address this challenge, the minimum-load path is computed for a single actuator which is responsible for the data gathering at a certain subregion. After the initial deployment of actuators to subregions, their workload is computed dynamically in light of the partition result of Voronoi diagrams. When an imbalance is identified, actuators are moved following prespecified rules to achieve a balanced load distribution. In fact, we are inspired by this technique to propose our clustering and adjustment strategies. Our technique leverages the grid cell division for avoiding irregular subregions, as presented in the evaluation part (Section VII-B). In this work, the authors divide the network region in terms of Voronoi diagrams, which may be irregular subregions in some cases. Therefore, the partition may not be optimal, especially when the path traversing energy consumption is unnegligible. Similar to the method presented in [9], the technique in [17] argues that no static sensors are deployed for sensing data in relatively large regions. Hence, mobile sensors are dispatched to visit critical sensing locations for gathering sensory data. The routes are planned almost the same in path length for ensuring the load balance of mobile sensors. Leveraging the same assumption that only mobile sensors, while no static sensors, exist in the network, the technique in [10] studies the region coverage challenge, especially when the coverage priority of different fields may be different. Generally, these techniques consider the energy consumption of mobile sink movement, while the energy comsumption of data gathering at certain subregions is assumed to be the equivalent. This assumption may not be valid in certain domain applications. Therefore, we propose a mobile sink scheduling method for balancing the energy consumption of mobile sinks, considering both data gathering and sink movement.

In quasi-real-time applications, information delay can be tolerant to a limited extent. In this setting, Gu *et al.* [6] propose to schedule one single mobile sink to travel over sink sites for gathering environmental variables with delay constraints. Delay is mainly caused by the movement of the mobile sink. To solve this issue, sink sites are determined such that a mobile sink can gather sensory data when traversing these sink sites only. The formalized mixed integer nonlinear programming problem is time consuming to be solved directly, and thus it is converted to tractable subproblems. In this research, the mobile sink is assumed to have limitless energy compared to static sensors. This assumption may not be held in certain domain applications. The authors surveyed the sink mobility management techniques in WSNs [27]. This survey is very interesting and has inspired us to develop the technique presented in this paper. Different from the above approach, the authors proposed in [7] a method to maximize the network lifetime for delay tolerant WSNs. Static sensors store sensory data temporarily and transmit to the mobile sink whenever it locates in a position most favorable for achieving the maximinum of network lifetime. The assumption of delay-tolerant applications is similar to what we have made in this paper. A similar technique is proposed in [18], where information is routed to a mobile sink in a multihop fashion. A distributed algorithm is proposed for prolonging the network lifetime under energy constraints through casting it into a linear program, and solving it through a dual decomposition method. Generally, these techniques may work well when one single mobile sink is considered, whereas the solution for supporting the cooperation of multiple mobile sinks is not explored.

In a task (or event)-driven network, sink scheduling is to assign tasks to potential sinks. The technique in [19] investigates the task assignment issue in a wireless sensor and robot network. When multiple tasks happen simultaneously, the assignment of the most appropriate robot to fulfill a certain task is a challenge. However, this work does not give solutions to remedy this issue. Sensor self-deployment is important, especially when the network size is large and the bandwidth is limited. Therefore, [20] studies this issue in a localized manner, where each sensor makes self-deployment decisions independently leveraging the information about k-hop neighborhoods. Two strictly localized algorithms are developed and they are proved efficient with desired coverage guarantee. [21] studies the mobile sensor relocation problem when sensors dropping or failure happens, for mitigating the coverage challenge. In summary, these approaches focus primarily on task prediction and assignment, as well as sensor relocation, for improving the region coverage. Balancing the energy consumption of mobile sinks, and thus to prolong the network lifetime, is not the main concern.

Besides, there are some techniques investigating similar challenge addressed in this paper. The authors propose in [22] an integer linear programming model for the optimization of sink locations and sensor-to-sink information flow routes between sensors and mobile sinks. [23] proposes to prolong the network lifetime through a controlled mobile sink by restricting the distance between two mobile sinks' sojourn location and the sojourn time. This work is improved through considering multiple mobile sinks [24]. Specifically, a WSN is divided into k clusters with a load-balanced tree. The routes and sojourn time of these k sinks are optimized in a heuristic manner. It is envisioned that thousands of independent components should be involved in an IoT application [25], which has a property called sparseness in the transformation process. Therefore, a desirable data compression ratio is important when considering the performance of applications. Hence, the authors propose a compressed sensing-oriented technique to explore the information acquisition in both IoT and WSNs. [26] proposes to apply mobile robots for supporting the real-time search and monitoring in remote sensing, where the coverage is the main concern. Based on the discussion above, we argue that current approaches have explored the scheduling of IoT mobile

sinks in a hybrid WSN. They have investigated the dispatch of mobile sinks with multicapability, in the sparse (or) static sensor deployment situation. However, scheduling mobile sinks efficiently, while prolonging the network lifetime, is a challenge not explored extensively. To fill this gap, this paper proposes a three-phase energy-balanced heuristic, where the energy consumption of data gathering and sink movement are taken into account, to balance the energy consumption of mobile sinks and to prolong the network lifetime.

IV. DIVIDING NETWORK REGION INTO GRID CELLS

Grid-based techniques have been used widely to support the applications of WSNs [13]. This section introduces the first step of our technique, which is to represent the network region as grid cells. Generally, these grid cells are equivalent in size and square in shape. The purpose is to facilitate the grid cell clustering and cluster adjustment procedures to be presented in the following sections. When this grid cell division is enacted, no update is necessary afterward. As argued in [27], a grid overlay may not be necessary when the number of sensor nodes deployed in the network is not large. In this technique, there is no assumption made about the network scale. Therefore, a grid overlay is adopted to represent the network region. The geographical size of a grid cell is determined by its side length gSide. Generally, a grid cell contains some static sensors, which are responsible for sensing environmental variables. There is (are) one (or multiple) position(s) in each grid cell, called the sink position(s), such that a mobile sink can gather data of all static sensors in this grid cell when staying at this (or these) sink position(s). The sink position(s) selection is affected by two parameters: 1) r; and 2) gSide. Note that r is fixed, and thus, gSide is the only deterministic factor.

1) When gSide is set to a relatively small value, the number of grid cells should be very large. Note that there is at least one sink position in each grid cell. This means that mobile sinks should spend much more energy on moving between grid cells within a cluster. For instance, gSide in Fig. 2 is set to half of that in Fig. 3, and we set gSide in Fig. 2 as one unit of distance. A mobile sink needs to move 31 units of distance in Fig. 2, while it has to move 14 units of distance in Fig. 3. Since gSide is set to $\sqrt{2}r$ in Fig. 3, a mobile sink can gather sensory data from all sensor nodes in the cluster when staying at the corresponding sink location (i.e., the center of a grid cell). This suggests that the energy to be consumed by the mobile sink for the case of Fig. 3 is sufficient, while the extra energy consumption for the movement of $(31 - 14 = 17)$ units of distance in Fig. 2 is unnecessary and can be avoided. From these two figures, it is evident that it is not energy efficient when gSide is set to a too smaller value than $\sqrt{2}r$. In addition, too small a side length of grid cells may induce hole or island problems, especially when static sensors are distributed in a skewed manner.
2) When gSide is set to a relatively large value, the network will be divided into some large grid cells. Hence, multiple sink positions are required for each grid cell and $g_i \cdot \mathrm{ECD}$ is a large value, although the energy consumption for the sink movement may be the same, or does not differ much, compared with that for the case when gSide is set to $\sqrt{2}r$. Therefore, after the initial grid cell clustering, a grid cell adjustment strategy considering the sink movement cost may be hard, if not impossible, to achieve a balanced situation, such that the deviation on energy consumption of all mobile sinks is within an allowed threshold.

Note that in [14], some experiments have been performed to study the optimal cluster size of WSNs, while optimizing the energy consumption of the network. It is evident that in most cases, the most optimal cluster size is where sensors are within the one-hop distance to the cluster head sensor (i.e., sink position). Leveraging this observation, gSide is set to a value that is not larger than $\sqrt{2}r$ and the sink position is set as the center of a grid cell in this paper, as illustrated in Fig. 3. Therefore, when a mobile sink moves to a sink position, all static sensors in the grid cell can forward their sensory data to this mobile sink. In this setting, static sensors do not require to communicate with each other and send data to the mobile sink in a hop-by-hop fashion. Based on this grid division strategy, we will present our grid cell clustering procedure, while considering the energy consumption of data gathering, in the next section.

V. GRID CELLS CLUSTERING CONSIDERING DATA GATHERING ENERGY CONSUMPTION

This section presents the second step of our technique, which is to group grid cells into clusters, such that in each cluster, a mobile sink should consume appropriately a similar amount of energy when gathering data from static sensors. Leveraging this grid cell clustering result, the energy consumption for the movement of mobile sinks is to be considered in Section VI through adjusting grid cells between clusters.

Without loss of generality and for simplicity, the number of clusters is set to the same number as that of mobile sinks. With this consideration, in this step, the number of static sensors contained in each cluster should not differ much after the grid cell clustering. We use a strategy inspired by the k-dimensional tree algorithm for clustering grid cells. Different from the k-dimensional tree algorithm, which clusters spatial data points through splitting the point set by their x- and y-coordinates, our technique clusters grid cells and a grid cell should be assigned into only one cluster.

As presented in Algorithm 1, we iterate the cell division procedure m times and generate one cluster in each iteration. It is worth mentioning that the division is guided by the variance of static sensors in grid cells along their x- or y-coordinate. A smaller variance is chosen in each round of division, which ensures that the two clusters generated after the division is much more similar in the number of static sensors contained.

First, the variances of static sensors with respect to their x- and y-coordinates are computed, respectively (lines 3–4). A larger value of the variance suggests that static sensors are distributed more evenly along their x- (or y-) coordinate. In this manner, it is highly possible that clusters generated along their x- (or y-) coordinate are similar in geographical size.

If the variance of x-coordinate is larger, the new cluster will be generated through the division along x-coordinate (lines

6–14). First, static sensors are sorted with respect to their x-coordinates (line 6), and the static sensor sn, which position is $\lceil e \rceil = 4$, is chosen as the position mark. Note that $\lceil e \rceil$ specifies the ceiling operation for e. For instance, $\lceil e \rceil = 4$ when $e = 3.34$. Static sensors contained in grid cells are numbered and counted while considering the cases of including (or excluding) the grid cell which contains sn (lines 8–9). Grid cells, which is in the upper and/or lower side(s) of the grid cell containing sn, are also being included (or excluded). These two numbers are encoded as cL and cR, respectively. A cluster is formed and includes grid cells containing static sensors in cL or cR. In a similar fashion, when the variance of y-coordinate is relatively larger, the same procedure is applied to generate a new cluster (lines 16–24).

Algorithm 1. GridCluster

Require:
G:the set of grid cells generated from a hybrid WSN S, where S contains n static sensors and m mobile sinks
Ensure:
$c_1 \dots c_m$: m clusters to be divided
1: **while** $m > 1$ **do**
2: $e \leftarrow$ (the number of static sensors in S) / m
3: $vX \leftarrow$ variance of x-coordinate of static sensors in S
4: $vY \leftarrow$ variance of y-coordinate of static sensors in S
5: **if** $vX \geqq vY$ **then**
6: $Q_x \leftarrow$ sort static sensors w.r.t. their x-coordinate
7: $sn \leftarrow$ take the static sensor which is in the position of $\lceil e \rceil$ in Q_x
8: $cL \leftarrow$ the number of static sensors whose x-coordinate is less than that of sn, while excluding the grid cell containing sn
9: $cR \leftarrow$ the number of static sensors whose x-coordinate is less than that of sn, while including the grid cell containing sn
10: **if** $|cL - e| \geqq |cR - e|$ **then**
11: $c_m \leftarrow$ a cluster containing a set of grid cells, such that these grid cells contain all static sensors in *cL*
12: **else**
13: $c_m \leftarrow$ a cluster containing a set of grid cells, such that these grid cells contain all static sensors in *cR*
14: **end if**
15: **else**
16: $Q_y \leftarrow$ sort static sensors w.r.t. their y-coordinate
17: $sn \leftarrow$ take the static sensor which is in the position of $\lceil e \rceil$ in Q_y
18: $cT \leftarrow$ the number of static sensors whose y-coordinate is less than that of *sn*, while including the grid cell containing *sn*
19: $cB \leftarrow$ the number of static sensors whose y-coordinate is less than that of *sn*, while excluding the grid cell containing *sn*
20: **if** $|cT - e| \geqq |cB - e|$ **then**
21: $c_m \leftarrow$ a cluster containing a set of grid cells, such that these grid cells contain all static sensors in *cT*
22: **else**
23: $c_m \leftarrow$ a cluster containing a set of grid cells, such that these grid cells contain all static sensors in *cB*
24: **end if**
25: **end if**
26: $m \leftarrow m - 1$
27: $G, S \leftarrow$ remove grid cells and static sensors in c_m
28: **end while**

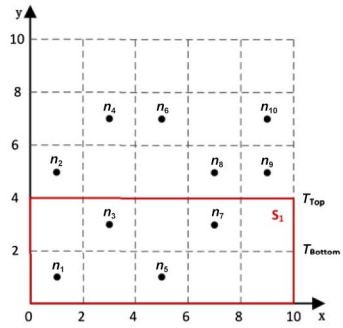

Fig. 4. Example demonstrating the new cluster generation procedure.

An example is shown in Fig. 4 illustrating the new cluster generation procedure. A hybrid WSN S contains three mobile sinks ($m = 3$) and 10 static sensors [denoted $n_1(1,1)$, $n_2(1,5)$, $n_3(3,3)$, $n_4(3,7)$, $n_5(5,1)$, $n_6(5,7)$, $n_7(7,3)$, $n_8(7,5)$, $n_9(9,5)$, and $n_{10}(9,7)$]. The side length of grid cells gSide is set to 2 units of distance, as shown in Fig. 4. In this setting, these 10 static sensors should be assigned into $m = 3$ clusters, and each cluster should contain $e = 3.34$ static sensors. In the first iteration, the variances of the x- and y-coordinates (vX and vY) are computed as 8 and 8.36, respectively. Static sensors are sorted along their y-coordinate from small to large in Q_y, and set the static sensor, which is in the position of $\lceil e \rceil = 4$ in Q_x, as sn. Thereafter, as marked in Fig. 4, T_{Top} is sn's nearest top grid side, and T_{Bottom} is sn's nearest bottom grid side. We get that $|\text{cT} - e| = 0.66$ is less than $|\text{cB} - e| = 1.34$. Hence, we set static sensors under T_{Top} as c_3 (as marked in the red rectangle in Fig. 4), remove static sensors in c_3 from S, and thereafter, a new cluster is generated accordingly.

In a nutshell, this section groups grid cells into clusters. This step tries to make the number of static sensors in each cluster as equivalent as possible, while can hardly guarantee that the bias of the geographical size of clusters is within a certain threshold. This is due to the fact that the number of static sensors is mainly considered when clusters are generated (refer to lines 11, 13, 21, 23 in Algorithm 1). This means that the energy consumption of mobile sinks may be significantly different when considering the energy consumption of sink movement, since mobile sinks need to traverse all grid cells that a certain cluster contains for gathering data from static sensors, and sink movement consumes the majority of energy [16]. To remedy this issue, we propose in Section VI to transfer grid cells between clusters, ensuring that clusters are similar in energy consumption considering the cost of data gathering and sink movement.

VI. CLUSTER ADJUSTMENT CONSIDERING MOBILE SINK MOVEMENT ENERGY CONSUMPTION

Grid cells have been grouped into clusters in Section V, where the energy consumption of data gathering is similar for clusters. However, clusters may differ significantly in the geographical size of their subregions, which induces the difference in the energy consumption of sink movement. To balance the whole energy consumption of mobile sinks, this section introduces our cluster adjustment strategy, which is to transfer grid cells between clusters, for achieving a relatively balanced energy consumption considering sink movement cost.

Intuitively, when a cluster is bigger in the number of static sensors and/or larger in geographical size, which should induce more energy consumption than average, one or more grid cells contained in this cluster should be transferred to its neighbor cluster(s) whose energy consumption is less than the average. Consequently, the energy consumption of all clusters is more balanced than that before the adjustment. As presented in Algorithm 2, clusters are examined in sequence (line 2), and grid cells are reallocated from a cluster to a neighboring one when necessary (lines 5 or 8). This procedure terminates when V (see Formula 1 in Section II) is relatively stable. Note that the situation where V repeats between several fixed values is considered as stable. In fact, this situation specifies the case that one (or several) grid cell(s) is(are) being realocated from a cluster to another cluster, being transferred back to the original cluster in the next round, and this procedure iterates afterward. It is worth mentioning that the criterion for the termination of Algorithm 2 (i.e., V is relatively stable) indicates that the bias of $c \cdot$ EC for mobile sinks can hardly be further decreased under certain network and parameter settings, which suggests that the network lifetime can hardly be prolonged further. As discussed, the network life is determined by the lifetime of the mobile sink, which is moving along a predetermined path to gather data in the cluster with the largest $c \cdot$ EC. The clusters generated with respect to the smallest V during the clustering procedure should be kept as the result of final grid cell division.

Algorithm 2. ClusterAdjustment

Require:
$c_1 \ldots c_m$: m clusters generated by Algorithm 1
Ensure:
$c_1 \ldots c_m$ are balanced in energy consumption
1: $E \leftarrow \frac{\sum_{i=1}^{m} ci.EC}{m}$
2: **while** V is *unstable* **do**
3: **for** $(i = 1; i <= m; i++)$ **do**
4: **if** $c_i.EC > E$ **then**
5: *DeclineCluster*(c_i, E)
6: **end if**
7: **if** $c_i.EC < E$ **then**
8: *IncreaseCluster*(c_i, E)
9: **end if**
10: **end for**
11: **end while**

Algorithm 3. DeclineCluster

Require:
c_i: the cluster whose grid cells need to be transferred to one of its neighbor clusters
E: average energy consumption for all clusters
Ensure:
one or several grid cells in the cluster c_i is(are) transferred to one of its neighbor cluster(s)
1: c_j: a cluster neighboring to c_i and has the smallest energy consumption considering data gathering and mobile sink movement
2: $k \leftarrow 1$
3: **while** $c_i.EC > c_j.EC$ **and** $c_i.EC > E$ **do**
4: $c_i.EC \leftarrow c_i.EC - gn_k(i,j).ECD - uECM$
5: $c_j.EC \leftarrow c_j.EC + gn_k(i,j).ECD + uECM$
6: $k \leftarrow k + 1$
7: **end while**
8: transfer $gn_1(j,i), \ldots, gn_k(j,i)$ from c_i to c_j

The function *DeclineCluster* is presented in Algorithm 3, which aims to transfer some grid cells contained in the cluster c_i to its neighboring cluster c_j, such that the difference between $c_i \cdot$ EC and $c_j \cdot$ EC should be reduced. Note that c_j represents one of $c_i's$ neighboring clusters which has the smallest energy consumption (line 1). Intuitively, $c_j \cdot$ EC should not be adjusted to a number of too large or too small compared with the average energy consumption of each cluster E. Otherwise, $c_j \cdot$ EC should further transfer grid cells to or from neighboring clusters, which should reduce the speed of clustering procedure. When $c_j \cdot$ EC is larger than $c_j \cdot$ EC and E, grid cell $\text{gn}_k(i,j)$ in the set $\text{GN}(i,j)$ is to be transferred from c_i to c_j. This procedure terminates until $c_i \cdot$ EC is not larger than either $c_j \cdot$ EC or E (lines 3–7). These grid cells are transferred from c_i to c_j finally (line 8). Since the function *IncreaseCluster* as presented in Algorithm 4 is similar to the function *DeclineCluster*, we do not discuss it in detail.

An example is shown in Fig. 5 about the grid cell adjustment procedure. The thick black line in the middle denotes the boundary between two clusters c_1 and c_2. The initial distribution generated by Algorithm 1 is shown in Fig. 5(a), where two clusters c_1 and c_2 are formed and the relation $c_1 \cdot \text{EC} > c_2 \cdot \text{EC}$ holds. Thus, the grid cells $\text{gn}_1(1,2)$ and $\text{gn}_2(1,2)$ are required to be transferred from c_1 to c_2. Thereafter $c_1 \cdot \text{EC} <= c_2 \cdot \text{EC}$ holds, and the transformation procedure ends. Note that the marks of grid cells are adjusted after the transformation procedure, as shown in Fig. 5(b), and only the grid cells next to the boundary in the initial distribution can be a $\text{GN}(i,j)$ cell. For instance in Fig. 5(a), grid cells 2, 3, 6, 7, 10, 11, 14, or 15 can be a GN(1,2) cell, while grid cells 1, 4, 5, 8, 9, 12, 13, or 16 cannot be.

When this grid cell adjustment strategy has been enacted, the difference of energy consumption for all clusters can hardly be reduced any further, considering the energy consumption of data gathering and sink movement. Consequently, the lifetime of mobile sinks is the longest somehow for current network and environmental settings. Therefore, the lifetime of the network is prolonged.

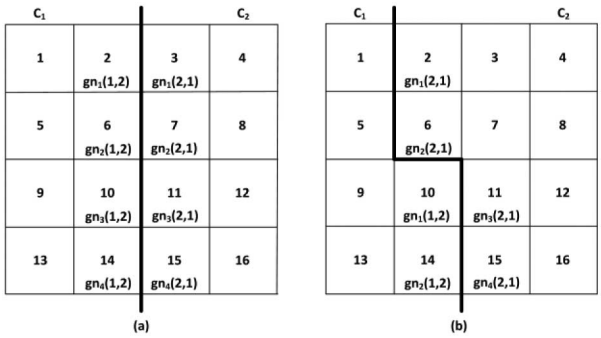

Fig. 5. Example illustrating the reallocation procedure of grid cells from the cluster c_1 to the neighboring cluster c_2.

Algorithm 4. InscreaseCluster

Require:

c_i: the cluster which needs to get grid cells from neighboring cluster(s)

E: average energy consumption in each cluster

Ensure:

Grid cells are added to c_i from a neighboring cluster

1: c_j: a cluster neighboring to c_i and has the biggest energy consumption considering data gathering and mobile sink movement

2: $k \leftarrow 1$

3: **while** $c_i.EC < c_j.EC$ **and** $c_i.EC < E$ **do**

4: $\quad c_j.EC \leftarrow c_j.EC - gn_k(j,i).ECD - uECM$

5: $\quad c_i.EC \leftarrow c_i.EC + gn_k(j,i).ECD + uECM$

6: $\quad k \leftarrow k + 1$

7: **end while**

8: transfer $gn_1(j,i), \ldots, gn_k(j,i)$ from c_j to c_i

VII. EXPERIMENTAL EVALUATION

A. Experiment Settings

The prototype has been implemented in a Java program. A hybrid WSN S is constructed, where $m = 6$ mobile sinks and $n = 10\,000$ static sensors are deployed in a region of $1000\,\text{m} \times 750\,\text{m}$ network space. We set $r = 50$ m and gSide $= 50$ m. gSide can be $50\ \text{m} \times \sqrt{2} = 71$ m of appropriate according to the discussion in Section IV. S is divided into $20 \times 15 = 300$ grids cells. As to the energy consumption of data gathering ($g_i \cdot$ ECD) and mobile sink movement (uECM), without loss of generality, we set $g_i \cdot$ ECD as 0.1 unit of energy when gathering data from one static sensor contained in a certain grid cell. $g_i \cdot$ ECD is proportional to the number of static sensors contained in a certain grid cell. uECM is set to 20 units of energy when moving a distance of 71 m. Note that the values for uECM and $g_i \cdot$ ECD can be set to other appropriate values when necessary. With this setting of parameters, we generated seven sets of datasets for static sensors with random distribution for the evaluation purpose. Specifically, 10 000 static sensors are randomly assigned to locations in the network region, and the distribution is uneven in most scenarios. Experiments are conducted on a laptop with an Intel i5-3210M processor, 8 GB memory, and a 64-bit Windows 8 operating system.

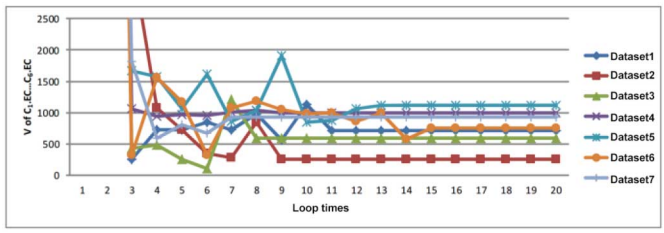

Fig. 6. Diversification of V on datasets $1, \ldots, 7$ when transferring grid cells between clusters.

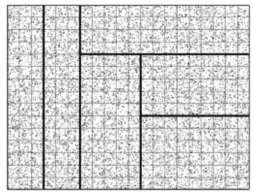

Fig. 7. Grid cell clustering result on dataset 7.

B. Evaluation Results

Three experiments have been performed for evaluating the convergence of our cluster adjustment strategy (as presented by Algorithm 2), the effectiveness of our grid cell transformation strategy, and the impact of gSide to clustering results.

Fig. 6 illustrates the diversification of V when Algorithm 2 is performed on datasets $1, \ldots, 7$, respectively. It is evident that V converges after several iterations, for instance, 11 iterations for dataset 1, and 9 iterations for dataset 2. This suggests that our technique presented in Algorithm 2 can generate an appropriate division of grid cells after limited iterations of (de)allocating grid cells between clusters, such that these clusters are similar (within an allowed divergence), considering the energy consumption of data gathering and sink movement between grid cells. The curves in Fig. 6 show that in each iteration, Algorithm 2 transfers grid cells from the clusters with local maximum energy consumption to the clusters of local minimum, in order to decrease V. Note that V may increase in some iterations, since the optimization of the local does not mean that of the global. On the other hand, when V is relatively stable (i.e., the fluctuation of V is within a certain threshold), the status of a global optimization in current network and parameter settings is achieved. Then, the cluster adjustment procedure finishes. The result of the clustering procedure corresponds to the clusters when V is the smallest, as mentioned in Section VI. For instance, Fig. 7 shows the clustering result on dataset 7 after the grid cells division (as detailed in Algorithm 1). Fig. 8 shows the cluster adjustment result after transferring certain grid cells between clusters for achieving a balance of energy consumption.

Note that we have presented in Fig. 1(c) and (d) that some special situations may cause $c_i \cdot$gnum steps or $c_i \cdot \text{gnum} + 1$ movement steps of mobile sinks for the traverse of grid cells in a certain cluster. These cases are evident in Fig. 8 and should be avoided whenever possible. Leveraging the analysis on our experimental results, it is found that the existence of these cases is related to the GN(i, j) grid cell transformation sequence in

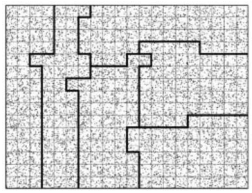

Fig. 8. Cluster adjustment result on dataset 7.

Table II
GN(i, j) GRID CELLS TRANSFORMATION STRATEGIES WHEN TRANSFERRING GRID CELLS FROM THE CLUSTER c_i TO THE NEIGHBORING CLUSTER c_j

scenario	transformation sequence
c_i is on c_j's left side	From lower left cell of GN (i,j)
c_i is on c_j's right side	From upper left cell of GN (i,j)
c_i is on c_j's top side	From lower right cell of GN (i,j)
c_i is on c_j's bottom side	From upper left cell of GN (i,j)

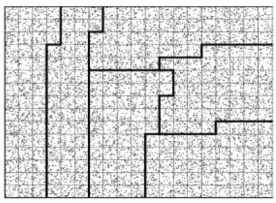

Fig. 9. Cluster adjustment result on dataset 7 using strategies in Table II.

Algorithm 2. Actually, GN(i, j) grid cells are transferred from $\text{gn}_1(i, j)$ to $\text{gn}_k(i, j)$ in sequence. To avoid the appearance of these cases, we have set some cell transformation strategies as presented in Table II. Fig. 9 shows the cluster adjustment result when these strategies are applied. In comparison with Fig. 8, the boundary of clusters is smoother in Fig. 9 when these strategies are applied, and the cases shown in Fig. 1(c) and (d) do not appear.

Fig. 10 shows the diversification of V on datasets $1, \ldots, 7$ when these strategies are adopted. Similar to the situation of not applying these strategies (as shown in Fig. 6), V converges after some iterations of grid cell adjustment procedures. Note that the number of iterations may increase (e.g., from 9 to 19 for dataset 2) or decrease (e.g., from 8 to 7 for dataset 3) when V converges. Fig. 10 shows that the V of the smallest is decreased (to a large extent) for most datasets against that as shown in Fig. 6. Note that V may become larger after the grid cells adjustment using the transformation strategies proposed in Table II, since static sensors may be in an appropriate distribution initially. In this case, the clusters with respect to the V of the smallest during the clustering procedure is the final result. An example is dataset 2, whose V in Fig. 10 is much larger than that in Fig. 6, and hence, the V at the ninth iteration as shown in Fig. 6 is to be adopted. We have examined and found that static sensors of dataset 2 are distributed in a more skewed fashion compared to the others. Specificcally, some regions are extremely dense, while the others are relatively sparse. In this setting, adjusting grid cells neighboring between clusters may induce the increase of V to an extent.

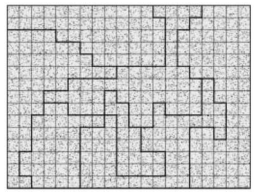

Fig. 10. Diversification of V of cluster adjustment results on datasets $1, \ldots, 7$ when transformation strategies presented in Table II are applied.

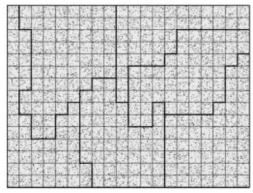

Fig. 11. Diversification of V on dataset 7 when gSide is set to different values.

We have discussed the impact of gSide on our clustering and cluster adjustment results as presented in Section IV. We have set six different values for gSide and conducted the experiments upon dataset 7. Note that when gSide is set to 80 or 100 m, four sink locations are set in each grid cell. The result is shown in Fig. 11. It is evident that when gSide is set to a relatively large value (e.g., 80 or 100 m as shown in Fig. 11), V can hardly converge within an allowed threshold. This means that an assignment task of grid cells to mobile sinks fails. On the other hand, when gSide is set to a relatively small value (for instance, 25 or 40 m as shown in Fig. 11), V can converge to a relatively small value very quickly (within 4 to 6 iterations). However, setting gSide to a smaller value cannot make V converge to a very small value.

C. Comparison With mTSP Mobile Sinks Scheduling

As mentioned above, the scheduling of mobile sinks can be reduced to the problem of the mTSP [11]. Therefore, experiments have been conducted to compare the performance of our technique with the mTSP algorithm. As shown in Fig. 12, when the network region is represented as grid cells, each grid cell g is represented using a node nd_{g1} central in this grid cell g (for instance, the node nd_{g1} in the top-left grid cell in Fig. 12). A weight nd_{g1}^{w} is set upon nd_{g1}, which corresponds to $g_1 \cdot \text{ECD}$, representing the energy to be consumed by a mobile sink when it stays at nd_{g1} and gathers data sensed by all static sensors in g. An edge ed connects nd_{g1} with a neighboring nd_{g2}, where nd_{g2} is the central node of a neighboring grid cell g_2 in the upper, lower, left, or right side of g, if applicable. A weight ed^{w} is set upon ed, which corresponds to uECM, representing the energy to be consumed by a mobile sink when moving from the node nd_{g1} to the node nd_{g2}.

Generally, mTSP is used when grid cells have been clustered into m clusters (denoted c_i where $i \in [1,\ m]$) as presented in Section V, and an improved TSP algorithm is applied in each cluster c_i for deriving a path p_{c_i} traversing all nd_g in this cluster c_i. It is worth mentioning that the number of grid cells and static

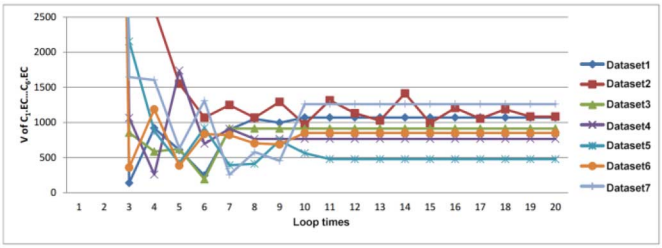

Fig. 12. Grid cells are represented by their central nodes for supporting the scheduling of mobile sinks using the mTSP algorithm.

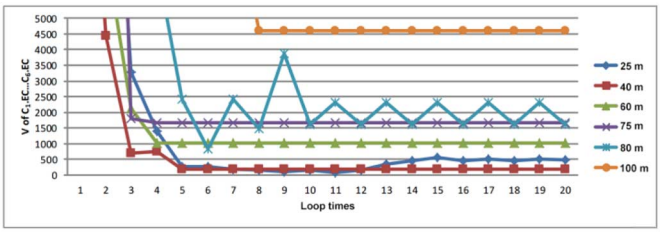

Fig. 13. Cluster adjustment result on dataset 7 for *Prin.*1.

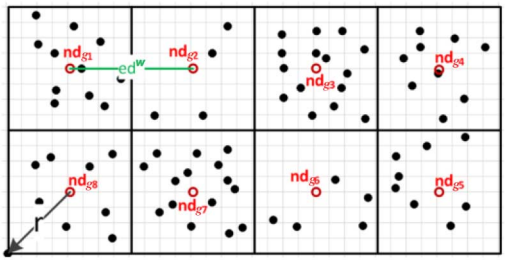

Fig. 14. Cluster adjustment result on dataset 7 for Prin. 2.

sensors contained in different clusters may be different. Hence, the energy consumption for data gathering and sink movement may differ significantly. Therefore, grid cells in the cluster c_i may be transferred from one cluster c_i to another c_j. Besides, unlike the traditional TSP algorithm, which finds a path starting and ending at a single node, a path starting and ending at different nodes is appropriate for our case, since the mobile sink can take the next round of traversal through the same path in the reverse direction. Specifically, after clustering grid cells as presented in Section V, a starting node $\mathrm{nd}_{\mathrm{init}}^{\mathrm{pth}}$ is determined for each cluster, which locates in the center of the corresponding cluster. Two kinds of principles are adopted for choosing the next node $\mathrm{nd}_g^{\mathrm{nt}}$, where the current node of the path is nd_g.

Prin. 1: Randomly, whenever there exists one node $\mathrm{nd}_g^{\mathrm{nt}}$ neighboring to nd_g and unoccupied.

Prin. 2, Besides Prin. 1, $\mathrm{nd}_g^{\mathrm{nt}}$ should be the nearest to $\mathrm{nd}_{\mathrm{init}}^{\mathrm{pth}}$ in the Euclidean distance.

As examples, the cluster adjustment results on dataset 7 by applying Prin. 1 and Prin. 2 are shown in Figs. 13 and 14, respectively. In comparison with Fig. 9, which shows our cluster adjustment strategy as presented in Section VI, Figs. 13 and 14 are much more irregular in shape.

Besides, experiments have been conducted to compare the difference in the energy consumption of mobile sinks for Prin. 1, Prin. 2, and our technique proposed in Section VI (denoted *ClsAdj* in Fig. 15), upon datasets 1 to 7, respectively. As to

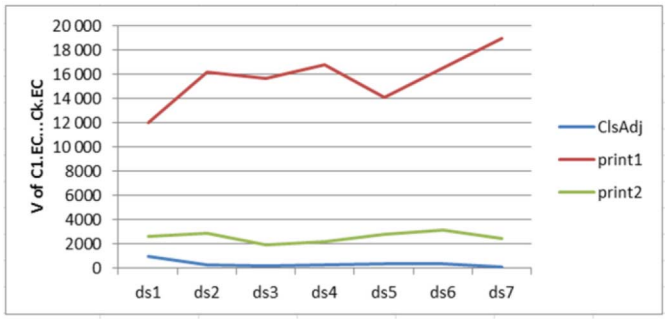

Fig. 15. Variance about the energy consumption of mobile sinks for Prin. 1, Prin. 2, and our technique *ClsAdj*.

the Prin. 1 and Prin. 2, for a certain path, the energy consumption includes $\sum nd_g^{\mathrm{w}}$, which represents the energy consumption for data gathering, and $\sum \mathrm{ed}^w$, which represents that for sink movement. Experiments have been conducted 10 times, and an average energy consumption is applied for computing their variances. As illustrated in Fig. 15, the variance in our technique (i.e., ClsAdj) is much smaller than those of Prin. 1 and Prin. 2. This means that for Prin. 1 and Prin. 2, the energy consumption of mobile sinks differ to a large extent. Therefore, some mobile sinks may deplete their energy much quicker than the others, and hence, the network lifetime is decreased. As to ClsAdj, the variance of energy consumption is quite small. This means that the energy consumption of mobile sinks does not differ much. Hence, the network lifetime for our technique is longer than that of Prin. 1 and Prin. 2.

VIII. CONCLUSION

Wirelessly connected IoT end nodes are changing the way we live and interact with the environment, and collaborative WSNs are integrated as a key component in the IoT architecture for creating novel pervasive smart environments [4]. Static WSNs are limited in achieving tasks for supporting certain domain applications. Using mobile sinks improve the capability of hybrid WSNs, where mobile sinks traverse along prespecified sink locations for gathering sensory data by static sensors. Prolonging the network lifetime, while ensuring the network region coverage, is a challenge. This paper proposed a method to mitigate this problem. Specifically, the network region is divided into grid cells, which are grouped into clusters while considering the energy consumption of data gathering. These clusters are adjusted through transferring grid cells among them, when considering the energy consumption of sink movement. Thereafter, mobile sinks are similar in energy consumption for both data gathering and sink movement. Consequently, the network lifetime is prolonged.

As to future work, as argued in [27], a prediction model can be adopted to support mobile sink management, when mobility patterns can be identified. This is a direction to be explored further. We have applied a grid overlay to represent the network region in this technique. This grid overlay may not be beneficial when the number of sensors is not large and the network region is not huge. Hence, a loosely coupled network structure may be appropriate to be used for exploring the mobile sink scheduling

problem. Besides, the network lifetime can be defined in other ways. The mobile sink management mechanism with various lifetime concern is one of future directions.

REFERENCES

[1] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[2] R. Piyare and S. R. Lee, "Towards Internet of Things (IoTs): Integration of wireless sensor network to cloud services for data collection and sharing," *Int. J. Comput. Netw. Commun.*, vol. 5, no. 5, pp. 59–72, 2013.

[3] C. Prabha, V. Ananthakrishnan, M. H. Supriya, and P. S. Pillai, "Localisation of underwater targets using sensor networks," *Int. J. Sens. Netw.*, vol. 13, no. 3, pp. 185–196, 2013.

[4] Q. Chi, H. Yan, C. Zhang, Z. Pang, and L. D. Xu, "A reconfigurable smart sensor interface for industrial WSN in IoT environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1417–1425, May 2014.

[5] H. Kim and B. K. Kim, "Online minimum-energy trajectory planning and control on a straight-line path for three-wheeled omnidirectional mobile robots," *IEEE Trans. Ind. Electron.*, vol. 61, no. 9, pp. 4771–4779, Sep. 2014.

[6] Y. Gu, Y. Ji, J. Li, and B. Zhao, "ESWC: Efficient scheduling for the mobile sink in wireless sensor networks with delay constraint," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1310–1320, Jul. 2013.

[7] Y. Yun, and Y. Xia, "Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, pp. 1308–1318, Sep. 2010.

[8] 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 Comput.*, vol. 13, no. 4, pp. 709–722, Apr. 2014.

[9] C. Wu, G. S. Tewolde, W. Sheng, B. Xu, and Y. Wang, "Distributed multi-actuator control for workload balancing in wireless sensor and actuator networks," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2462–2467, Oct. 2011.

[10] H. Mahboubi, J. Habibi, A. G. Aghdam, and K. Sayrafian-Pour, "Distributed deployment strategies for improved coverage in a network of mobile sensors with prioritized sensing field," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 451–461, Feb. 2013.

[11] H. Salarian, K.-W. Chin, and F. Naghdy, "An energy-efficient mobile-sink path selection strategy for wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2407–2419, Jun. 2014.

[12] H. Mahboubi, K. Moezzi, A. G. Aghdam, K. Sayrafian-Pour, and V. Marbukh, "Distributed deployment algorithms for improved coverage in a network of wireless mobile sensors," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 163–174, Feb. 2014.

[13] I.-H. Peng and Y.-W. Chen, "Energy consumption bounds analysis and its applictions for grid based wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 444–451, 2013.

[14] A. Forster, A. Forster, and A. L. Murphy, "Optimal cluster sizes for wireless sensor networks: An experimental analysis," in *Proc. Int. Conf. Ad Hoc Netw.*, 2010, pp. 49–63.

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

[16] 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 Distrib. Syst.*, vol. 21, no. 12, pp. 1836–1850, Dec. 2010.

[17] T.-L. Chin and Y.-T. Yen, "Load balance for mobile sensor patrolling in surveillance sensor networks," in *Proc. IEEE Int. Conf. Wireless Commun. Netw.*, 2012, pp. 2168–2172.

[18] M. Gatzianas and L. Georgiadis, "A distributed algorithm for maximum lifetime routing in sensor networks with mobile sink," *IEEE Trans. Wireless Commun.*, vol. 7, no. 3, pp. 984–994, Mar. 2008.

[19] I. Mezei, V. Malbasa, and I. Stojmenovic, "Task assignment in wireless sensor and robot networks," in *Proc. IEEE Int. Conf. Telecommun. Forum*, 2012, pp. 596–602.

[20] X. Li, H. Frey, N. Santoro, and I. Stojmenovic, "Strictly localized sensor self-deployment for optimal focused coverage," *IEEE Trans. Mobile Comput.*, vol. 10, no. 11, pp. 1520–1533, Nov. 2011.

[21] X. Li, G. Fletcher, A. Nayak, and I. Stojmenovic, "Randomized carrier-based sensor relocation in wireless sensor and robot networks," *Ad Hoc Netw.*, vol. 11, no. 7, pp. 1951–1962, 2013.

[22] E. Guney, N. Aras, I. Altinel, and C. Ersoy, "Efficient integer programming formulations for optimum sink location and routing in heterogeneous wireless sensor networks," *Comput. Netw.*, vol. 54, no. 11, pp. 1805–1822, 2010.

[23] W. Liang, J. Luo, and X. Xu, "Prolonging network lifetime via a controlled mobile sink in wireless sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, 2010, pp. 1–6.

[24] W. Liang and J. Luo, "Network lifetime maximization in sensor networks with multiple mobile sinks," in *Proc. IEEE Conf. Local Comput. Netw.*, 2011, pp. 350–357.

[25] S. Li, L. D. Xu, and X. Wang, "Compressed sensing signal and data acquisition in wireless sensor networks and Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2177–2186, Nov. 2013.

[26] Y. Pei and M. W. Mutka, "Stars: Static relays for remote sensing in multi-robot real-time search and monitoring," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 10, pp. 2079–2089, Oct. 2013.

[27] Y. Gu, F. Ren, Y. Ji, and J. Li, "The evolution of sink mobility management in wireless sensor networks: A survey," *IEEE Commun. Surv. Tuts*, doi: 10.1109/COMST.2015.2388779, Jan. 2015.

ZhangBing Zhou (M'15) received the Ph.D. degree in computer science from Digital Enterprise Research Institute (DERI), Galway, Ireland, in 2010.

He worked as a Software Engineer with Huawei Tech. Company, Ltd., Beijing, China, for 1 year and served as a Member of the Technical Staff at Bell Laboratories, Lucent Technologies (now Alcatel–Lucent), Beijing, for 5 years. Currently, he is a Full Professor with the University of Science and Technology, Beijing, and as an Adjunct Associate Professor at TELECOM SudParis, Evry, France. He has authored over 90 referred papers. His research interests include process-aware information system and sensor network middleware.

Dr. Zhou has served as an Associate or Guest Editor of more than 10 journals.

Chu Du received the Ph.D. degree in computer science from the China University of Geosciences, Beijing, China, in 2015.

Currently, he is a Senior Engineer with the 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, China. He has authored 10 referred papers. His research interests include sensor network middleware and process-aware information system.

Lei Shu (S'08–M'14) received the Ph.D. degree in computer science from Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, Ireland, in 2010.

Currently, he is with Guangdong University of Petrochemical Technology, Maoming, China, as a Full Professor. He has authored over 200 papers. His research interests include industrial sensor network middleware and process-aware information system.

Dr. Shu is an Editor-in-Chief for *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, and a member of ACM.

Gerhard Hancke (S'00–M'08–SM'11) received the D.Eng. degree in computer science from the University of Pretoria, Pretoria, South Africa, in 1983.

Currently, he is with the University of Pretoria, as a Professor, and the Chair of the Computer Engineering Program and Director of the Centre for Advanced Sensor Networks, a joint initiative between the Department of Electrical, Electronic, and Computer Engineering and the Meraka Institute at the Council for Scientific and Industrial Research. His research interests include distributed sensors and actuators networks.

Prof. Hancke is a Senior AdCom Member and a Secretary of the IEEE Industrial Electronics Society.

Jianwei Niu (M'07–SM'13) received the Ph.D. degree in computer science from Beihang University (BUAA), Beijing, China, in 2002.

He was a Visiting Scholar at Carnegie Mellon University, Pittsburgh, PA, USA, from January 2010 to February 2011. Currently, he is a Professor with the School of Computer Science and Engineering, BUAA. He has authored more than 100 referred papers, and filed more than 30 patents in mobile and pervasive computing. His research interests include mobile and pervasive computing.

Dr. Niu has served as an Associate Editor of the *International Journal of Ad Hoc and Ubiquitous Computing*, and the *Journal of Internet Technology*, and an Editor of the *Journal of Network and Computer Applications*.

Huansheng Ning (M'10–SM'12) received the Ph.D. degree in electrical engineering from Beihang University, Beijing, China, in 2001.

He is a Professor with the School of Computer and Communication Engineering, University of Science and Technology, Beijing. He has authored more than 50 papers in journals, international conferences, and workshops. His research interests include Internet-of-Things, aviation security, electromagnetic sensing, and computing.