DAO: Overcoming Overall Storage Overflow in Base Station-less Sensor Networks Via Energy-Efficient Data Aggregation

Bin Tang, Yan Ma, and Basil Alhakami
Computer Science Department
California State University Dominguez Hills
btang@csudh.edu, {yma4,balhakami1}@toromail.csudh.edu

***Abstract*—Many emerging sensor network applications operate in challenging environments, wherein it is not feasible to install a long-term base station with power plug to collect data. Data generated from such *base station-less sensor networks* therefore must be stored inside the networks for some unpredictable period of time before uploading opportunities become available. Consequently, generated data could overflow limited storage capacity available in the entire network, making discarding valuable data inevitable. To overcome such *overall storage overflow* in base station-less sensor networks, we propose and study a new algorithmic problem called *data aggregation for overall storage overflow (DAO)*. Utilizing the spatial data correlation that commonly exists among sensor data, DAO employs data aggregation techniques to reduce the overflow data size, while minimizing the total energy consumption and preserving as much information as possible. With proper graph transformation, we show that DAO is equivalent to *multiple traveling salesman walks problem (MTSW)*, a new graph-theoretic problem that has not been studied. We prove that MTSW is NP-hard. We then solve it optimally in linear topologies and design a $(2-\frac{1}{q})$-approximation algorithm for general graph topologies, where q is the number of nodes to visit (i.e., the number of sensor nodes that aggregate their overflow data). We further put forward a novel heuristic algorithm and empirically show that it constantly outperforms the approximation algorithm by $15\% - 30\%$ in energy consumption. Finally, we design a distributed data aggregation algorithm that can achieve the same approximation ratio as the centralized algorithm, while incurring comparable energy consumption.**

Keywords – Sensor Networks, Data Aggregation, Energy-Efficient Algorithms, Graph Theory

I. Introduction

With the advance in sensor technology and maturity of sensor network design and deployment, scientists are ready to utilize sensor networks to explore the physical world in a scope and a depth that was never reached before. For example, sensor networks are being designed and deployed in recent years to address some of the most fundamental problems facing human beings, such as disaster warning, climate change, and renewable energy. The emerging sensor networks designed for those scientific applications include seismic sensor networks [41], underwater or ocean sensor networks [4, 12], wind and solar harvesting [17, 33], and volcano eruption monitoring and glacial melting monitoring [31, 42]. One common characteristic of these sensor networks is that they are all deployed in challenging environments such as in remote or inhospitable regions, or under extreme weather, to continuously collect large volumes of data for a long period of time.

In such inaccessible sensor fields, it is not possible to deploy high-power, high storage data-collecting base stations with power outlets. Therefore sensory data generated have to be stored inside the network for some unpredictable period of time and then being collected by periodic visits of robots or data mules [15], or by low rate satellite link [32]. We refer to such sensor networks as *base station-less sensor networks*. Due to inadequate human intervention in the inhospitable environments, base station-less sensor networks must operate much more resiliently than traditional sensor networks (with base stations and in friendly environments).

In this paper, we focus on sensor network resilience against storage overflow of sensor nodes, wherein storage spaces of some sensor nodes are depleted and therefore it cannot store any newly generated data. Storage overflow is a major obstacle existing in above emerging sensor networks, due to following reasons. On one side, massive amounts of data in above scientific applications are generated, sensing a wide range of physical properties in real world ranging from solar light to wind flow to seismic activity. On the other side, storage is still a serious resource constraint of sensor nodes, despite the advances in energy-efficient flash storage [34] with good compression algorithms (data is compressed before stored) and good aging algorithms (fidelity of older data is reduced to make space for newer data). As a consequence, the massive sensory data could soon overflow data storage of sensor nodes and causes data loss. For example, according to [28], an acoustic sensor that has a 1GB flash memory and is designed to sample the entire audible spectrum will run out of its storage in just seven hours. Such storage overflow problem is further exacerbated in base station-less sensor networks, wherein the high-power, high-storage base stations are not available to collect and store the data most of the time. Below we give a more concrete example contributing to overall storage overflow in base station-less sensor networks.

Motivating Example. In an underwater sensor network monitoring coral reefs and fisheries [40], the wireless sensor nodes

have variety of sensing capabilities including cameras, water temperature, and pressure. An autonomous underwater vehicle (AUV) is dispatched periodically to upload the collected data. Each sensor has 512KB of flash memory, whereas a single 255×143 image requires 36KB of data storage. Therefore the storage may be filled quickly even if the images are taken at a very slow rate. To alleviate this problem, nodes with depleted storages can offload their *overflow data*, the part of newly generated data that can no longer be stored locally, to neighboring nodes with available storage before AUV arrives.

A more serious situation arises, however, when the total size of the overflow data exceeds the total available storage in the network, which obviously cannot be solved by aforesaid data offloading. We refer to this problem as *overall storage overflow* in sensor networks. Suppose in above scenario there are 100 nodes and 10 of them are generating one image per second, it just takes less than three hours to reach overall storage overflow. If the AUV cannot be dispatched timely due to inclement and stormy weather, and no other appropriate actions are taken, discarding valuable data becomes inevitable. Even using the latest parallel NAND flash technology with 16GB [22] and taking typical 640×480 JPEG color images, it takes less than one day to reach overall storage overflow. In this paper, we endeavor to answer below question: *For base station-less sensor networks that operate in challenging environments where human intervention is impossible, how to retain all the generated information notwithstanding the overall storage overflow?*

In this paper, we take advantage of spatial correlation that commonly exists among sensory data, and employ data aggregation techniques to overcome overall storage overflow. We formulate a new graph-theoretic problem called *data aggregation for overall storage overflow (DAO)*. To solve DAO, we design a suite of energy-efficient optimal, approximation, heuristic, and distributed data aggregation algorithms, with detailed analytical analysis of their performance guarantees. The novelty of our aggregation techniques is a routing structure called *minimum q-edge forest*, where q is the number of sensor nodes that aggregate their overflow data. We show that the minimum q-edge forest generalizes minimum spanning tree, one of the most fundamental data structures, and accurately captures information needed for energy-efficient data aggregation. At the core of DAO is a new graph-theoretic problem called *multiple traveling salesman walks (MTSW)*, which has not been studied in any of the existing work.

After being aggregated to the size accommodable by the network, the overflow data can then be stored at nodes with available storage using data offloading techniques proposed in [37] (we further illustrate this using Example 1 in Section II). Note that in this paper we do not consider how to upload data from sensor nodes to base station, which has been studied extensively by using data mules or mobile data collectors [24, 30].

The main contributions of this paper include the following.

1). We identify an overall storage overflow problem in sensor networks, and formulate a new data aggregation problem called DAO. (Section I and II)

2). We show that DAO is equivalent to a new *multiple traveling salesman walks problem (MTSW)* with appropriate graph transformation. (Section IV)

3). We prove that the MTSW is NP-hard. We solve it optimally in linear topologies and design a $(2 - \frac{1}{q})$-approximation algorithm for general graph topologies, where q is the number of nodes to visited or the number of aggregators in DAO. (Section III)

4). We design an efficient heuristic algorithm that further improves the approximation algorithm by $15\% - 30\%$ in terms of energy consumption. (Section VI)

5). We design a distributed data aggregation algorithm that achieves the same approximation guarantee as the approximation algorithm, while being optimal with regards to the message complexity. (Section V)

The rest of the paper is organized as follows. In Section II, we formulate the DAO problem and illustrate it with an example. In Section III we formulate the MTSW problem and present a suite of optimal, approximation, and heuristic algorithms. We show in Section IV that with proper graph transformation the DAO is equivalent to the MTSW therefore the algorithms for MTSW can be applied to solve DAO. In Section V, we design a distributed data aggregation algorithm for the DAO with time and message analysis, and show that it can achieve the same aggregation cost as the approximation algorithm does. In Section VI, we compare all the different algorithms under different network dynamics and discuss the simulation results. Section VII discusses state-of-the-art and the significance of our work. Section VIII concludes the paper with possible future research.

II. Data Aggregation For Overall Storage Overflow (DAO)

In this section, we first introduce the DAO with a problem statement. We then present its network model, data spatial correlation model, and energy model. We finally formally formulate the DAO and end with an illustrative example.

Problem Statement. Fig. 1 shows a base station-less sensor network. Some sensor nodes are close to the events of interest thus are constantly generating sensory data, depleting their own storages. We refer to sensor nodes with depleted storage spaces while still generating data as **data nodes**. The newly generated data that can no longer be stored at data nodes is called **overflow data**. To avoid data loss, overflow data is offloaded to sensor nodes with available storages (referred to as **storage nodes**).[1] To start the aggregation process, one or more data nodes (called **initiators**) send their overflow data to visit other nodes. When a data node receives the data, it becomes an **aggregator** by aggregating its own overflow data, and then forwards the initiator's entire overflow data, in multi-hop manner, to another data node. This data node becomes an aggregator and aggregates its overflow data, and so on and so

[1] Sensor nodes that have generated data but have not depleted their storage spaces are storage nodes, since they can store overflow data from data nodes.

forth. When a storage node receives the data, it simply relays it (note that a storage node cannot store part or entirety of the received data until data aggregation process is done, since other aggregators need the entire data in order to aggregate their own). This continues until enough aggregators are visited such that total size of the overflow data after aggregation equals to or is slightly less than total available storage in the network.

Any node participating in this process (including initiator, aggregator, and relaying storage node) consumes its own battery energy. *To save energy consumption, the challenge is therefore to select initiators from the data nodes, and to decide the sequence of aggregators/storage nodes visited by each initiator's data, such that enough number of aggregators are visited while costing minimum amount of energy.*

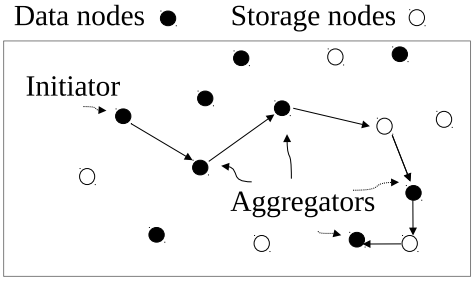


Fig. 1. An illustration of DAO.

Network Model. The sensor network is represented as an undirected connected graph $G(V, E)$, where $V = \{1, 2, ..., |V|\}$ is the set of $|V|$ sensor nodes, and E is the set of $|E|$ edges. There are p data nodes, denoted as V_d (the other $|V| - p$ nodes are storage nodes). Let R denote the size of generated overflow data in bits at each data node, and let m be the available storage space in bits at each storage node. (We leave that data nodes have different sizes of overflow data and storage nodes have different sizes of storage spaces as future work.) Due to the overall storage overflow, $p \times R > (|V| - p) \times m$, giving that $p > \frac{|V|m}{m+R}$ and $p \in \mathbb{Z}^+$.

Spatial Correlation Data Model. We propose an entropy-based spatial correlation model that is based upon [10]. The spatial correlation data model in [10] assumes that each sensor node generates a R-bit packet and transmits it back to the base station, therefore every node serves as an initiator. In our case, R is the amount of overflow data generated only at data node and only selected dada nodes can serve as initiators. Let $H(X)$ denote the entropy of a discrete random variable X, and $H(X|Y)$ denote the conditional entropy of a random variable X given that random variable Y is known. If data node i receives no side information from other data nodes, its overflow data is entropy coded with $H(i|j_1, ..., j_p) = R$ bits. If data node i receives side information from at least another data node, the size of its overflow data is $H(i|j_1, ..., j_p) = r \leq R$, $j_k \in V_d \wedge j_k \neq i, 1 \leq k \leq p$. This correlation model has two advantages. First, it captures the uniform data spatial correlation scenario, wherein data generated at different data nodes have similar correlation with each other (we leave the more challenging and realistic model that different nodes have different data correlation as future work). Second, it is an effective distributed coding strategy, which works well in large scale sensor network applications. We are aware of other distributed coding techniques such as Slepian-Wolf coding [9, 35]. However, they need global correlation structure, which is impractical in large scale sensor networks.

Based on this model, we have two observations about the data aggregation in DAO:

Observation 1: Each data node can be either an initiator, or an aggregator, or none of them, but not both of them. An initiator cannot be an aggregator because its data serves as side information for other nodes to aggregate. An aggregator cannot be an initiator since its aggregated data loses the side information needed for others nodes' aggregation. □

Observation 2: Each aggregator can be visited multiple times by the same or different initiators (if that is more energy-efficient). However, the data of an aggregator can only be aggregated once, with size reduced from R to r. Besides, when an initiator A is visited by another initiator B, it is equivalent to that B visits A and all other data nodes visited by A. □

Energy Model. We adopt first order radio model [13] for battery power consumption. When node u sends R-bit data to its one-hop neighbor v over distance $l_{u,v}$, transmission energy cost at u is $E_t(R, l_{u,v}) = E_{elec} \times R + \epsilon_{amp} \times R \times l_{u,v}^2$, receiving energy cost at v is $E_r(R) = E_{elec} \times R$. Here, $E_{elec} = 100nJ/bit$ is energy consumption per bit on transmitter and receiver circuits, and $\epsilon_{amp} = 100pJ/bit/m^2$ is energy consumption per bit on transmit amplifier. Let $W = \{v_1, v_2, ..., v_n\}$ be a *walk*, a sequence of n nodes with $(v_i, v_{i+1}) \in E$ and $v_1 \neq v_n$ (if all nodes in W are distinct, W is a *path*). Let $w(R, u, v) = E_t(R, l_{u,v}) + E_r(R)$, and $c(R, W) = \sum_{i=1}^{n-1} w(R, v_i, v_{i+1})$ denote *aggregation cost*, the total energy consumption of sending R-bit from v_1 to v_n along W. We assume that there exists a contention-free MAC protocol to avoid overhearing and collision (e.g. [6]).

Valid Range of p. We have calculated the lower bound of p. Next we compute the upper bound of p, so that the overflow data after aggregation can fit in the available storage. Let q denote the number of aggregators to visit. Since each aggregator reduces its overflow data size by $(R - r)$, and the total anticipated data size reduction is $p \times R - (|V| - p) \times m = p \times (R + m) - |V| \times m$, therefore all together

$$q = \lceil \frac{p \times (R + m) - |V| \times m}{R - r} \rceil \tag{1}$$

distinct aggregators need to be visited. Meanwhile, since at least one data node needs to be the initiator to start the aggregation process, there can only be maximum of $p - 1$ aggregators (Observation 1). We therefore have $\lceil \frac{p \times (R+m) - |V| \times m}{R-r} \rceil \leq p - 1$, which gives $p \leq \lfloor \frac{|V|m - R + r}{m + r} \rfloor$. The valid range of p is therefore

$$\frac{|V|m}{m + R} < p \leq \lfloor \frac{|V|m - R + r}{m + r} \rfloor. \tag{2}$$

Problem Formulation of DAO. Given a valid p value and its corresponding q value, meaning q of the p data nodes are aggregators, then the rest $p-q$ data nodes can serve as initiators (Observation 1). DAO selects:

- set of a ($1 \leq a \leq (p-q)$) initiators, denoted as $\mathcal{I}$, and
- corresponding set of a *aggregation walks*: W_1, W_2, ..., W_a, where W_j ($1 \leq j \leq a$) starts from a distinct initiator $I_j \in \mathcal{I}$, and $|\bigcup_{j=1}^{a}\{W_j - \{I_j\} - G_j\}| = q$. Here, G_j is the set of storage nodes in W_j thus $W_j - \{I_j\} - G_j$ is the set of aggregators in W_j. Since an aggregator can appear multiple times in the same or different aggregation walks (Observation 2), $\bigcup_{j=1}^{a}\{W_j - \{I_j\} - G_j\}$ signifies a set of q $distinct$ aggregators in the network.

That is, each initiator sends its overflow data to all the aggregators in its aggregation walk, such that all together q distinct aggregators are visited in the network, and the total energy consumption in this process $\sum_{1\leq j\in a} c(R, W_j)$, referred to as *total aggregation cost*, is minimized.

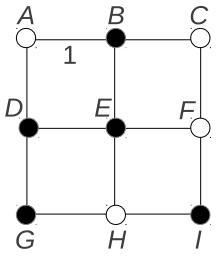


Fig. 2. A DAO example.

EXAMPLE 1: Fig. 2 gives an example of DAO in a grid sensor network of 9 nodes (we use grid only for illustration purpose – the DAO and its solutions are designed for general graph topologies). Nodes B, D, E, G, and I are data nodes, while A, C, F and H are storage nodes. We assume that $R = m = 1, r = 3/4$, and energy consumption along any edge is 1 for one unit of data. Overall storage overflow exists in this scenario, since there are 4 units of storage space while there are 5 units of overflow data. Using Equation 1, the number of aggregators q is 4, leaving one data node to be initiator. One of the optimal solutions could be selecting B as initiator and setting its aggregation walk as: B, E, D, G, H, I, with total aggregation cost of 5. □

Data Offloading After Data Aggregation. In Fig. 2, after aggregation, the sizes of overflow data at B, E, D, G, and I are 0, 3/4, 3/4, 3/4, and 7/4, respectively, totaling 4 units. Therefore they can be offloaded and stored into the 4 units of storage space available in the network. Note that 7/4 units of data at I now include 3/4 unit of I's own overflow data and one unit of B's overflow data. Next is to decide how to offload those data to storage nodes with minimum energy consumption.[2] This has been shown to be a minimum cost flow problem [37], which can be solved optimally and efficiently [1]. One optimal minimum cost flow solution is offloading E's 1/4 unit of data to A (cost 1/2), E's 2/4 unit of data to C (cost 1), D's 3/4 unit of data to A (cost 3/4), G's 3/4 unit of data to H (cost 3/4), I's 1/4 unit of data to H (cost 1/4), I's 2/4 unit data to C (cost 1), and B's one unit of data, now located at I, to F (cost 1), totaling 5.25 offloading cost.

Since data offloading can be achieved optimally, we only focus on data aggregation in this paper. We find that DAO gives rise to a new graph-theoretic problem, which we refer to as *multiple traveling salesman walks problem (MTSW)*. We formulate and solve MTSW in Section III. In Section IV, we show that the DAO is equivalent to the MTSW therefore the algorithms for MTSW can be applied to solve DAO.

III. Multiple Traveling Salesman Walks Problem (MTSW)

In this section, we first formulate the MTSW problem and prove its NP-Hardness. Then we solve MTSW by presenting an efficient optimal algorithm for linear topologies and an efficient $(2-\frac{1}{q})$-approximation algorithm for general graph topologies, respectively.

A. Problem Formulation and NP-Hardness.

Given an undirected weighted graph $G = (V, E)$ with $|V|$ nodes and $|E|$ edges, a cost metric (which represents the distance or traveling time between two nodes), the objective of the MTSW is to determine a subset of *at most* $b < |V|$ starting nodes (i.e., the initiators in DAO), from each of which a salesman can be dispatched to visit a number of other nodes (i.e., the aggregators in DAO) following a walk, such that a) all together $q = |V| - b$ nodes (excluding starting nodes) are visited, and b) the total cost of the walks is minimized.

Let $w(u, v)$ denote weight of edge $(u, v) \in E$. We assume that edge weights satisfy triangle inequality: for any three edges $(x, y), (y, z), (z, x) \in E$, $w(x, y) + w(y, z) \geq w(z, x)$. Given a walk $W = \{v_1, v_2, ..., v_n\}$, let $c(W) = \sum_{i=1}^{n-1} w(v_i, v_{i+1})$ denote the cost of traversing along W. The objective of MTSW is to decide:

- the set of a ($1 \leq a \leq b$) starting nodes $\mathcal{I} \subset V$, and
- the set of a *walks* $W_1, W_2, ..., W_a$: W_j ($1 \leq j \leq a$) starts from a distinct node $I_j \in \mathcal{I}$, and $|\bigcup_{j=1}^{a}\{W_j - \{I_j\}\}| = q$,

such that *total cost* $\sum_{1\leq j\in a} c(W_j)$ is minimized.

Theorem 1: The MTSW is NP-hard.

Proof: We show that traveling salesman walk problem (TSW) [20] is a special case of MTSW. TSW is defined as follows: Given a weighted graph, a starting node s and an ending node t, the goal is to find a minimum-length s-t walk that visits all nodes *at least* once. TSW is NP-hard (Section 6, [20]). The differences between TSW and MTSW are a) there can be multiple starting and ending nodes in MTSW while there is only one starting node s and one ending node t in TSW, and b) it needs to find the starting and ending nodes in MTSW whereas in TSW s and t are fixed. In the special case when $b = 1$, MTSW can be solved by calling TSW as a

[2]Currently data aggregation and data offloading are separated stages – we leave integrating them for a more energy-efficient solution as future work.

sub-routine upon all possible pairs of s and t, and finding the one that yields the minimum cost. Therefore TSW is a special case of MTSW and MTSW is NP-hard. ∎

B. Optimal Algorithm for Linear Topologies.

Linear sensor networks [16] have been well adopted in applications such as water pollution monitoring along the river bank and underwater seismic monitoring along the seashore. A linear sensor network is represented by a linear graph $G(V, E)$, which consists of $|V|$ nodes: 1, 2, ..., $|V|-1$, and $|V|$ from left to right. Two adjacent nodes u and v are connected by an edge, with weight $w(u, v)$. Algorithm 1 below works as follows. First it finds the q edges in E with smallest weights (line 2). Then it checks if any of pair of them are adjacent with a common node (lines 5-9). If so, they are merged into one path (in linear topologies, all the obtained walks are paths). Finally, for each path, it starts with the leftmost node of the path and visits the rest nodes in this path exactly once. Fig. 3 shows an example of a linear network of eight nodes, with weight of each edge shown on the top of each edge. Suppose $q = 4$; that is, it needs to find paths along which 4 nodes are visited with minimum cost. Algorithm 1 yields two paths: one is starting with node 2 and visiting nodes 3, 4, and 5, the other is starting with 6 and visiting 7, with total cost of 10. This is the minimum cost to visit 4 nodes.

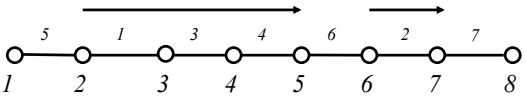
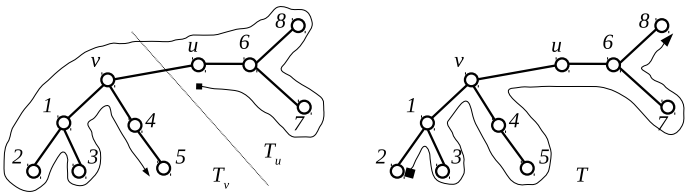


Fig. 3. An linear network of eight nodes with $q = 4$. The weight of each edge is shown on the top of each edge.

Algorithm 1: Optimal Algorithm For Linear Topologies.
Input: A linear topology $G(V, E)$ and q, number of nodes to visit;
Output: set of a paths: $W_1, W_2, ..., W_a$, $\sum_{1 \le j \in a} c(W_j)$;
0. **Notations**:
i: index for edges; j: index for paths;
W_j: j^{th} path obtained;
1. $i = 1$; $j = 1$;
2. Find the q smallest-weight edges, name them e_1, e_2, ..., e_q from left to right in linear topology;
$L(i)$, $R(i)$: left and right end node of edge e_i;
3. **while** $(i \le q)$
4. $W_j = \phi$ (empty set);
5. $I_j = L(i)$; $W_j = \{L(i), R(i)\}$;
6. **while** $(i < q \wedge R(i) == L(i+1))$
7. $W_j = W_j \cup \{R(i+1)\}$;
8. $i++$;
9. **end while;**
10. $i++$; $j++$;
11. **end while;**
12. $a = j$;



(a) B-Walk. (b) LP-Walk.

Fig. 4. Illustrating a walk in a tree T, in which $w(u, v) = 2$ and weights of other edges being 1. (a) shows a Binary Walk (B-walk): u, 6, 7, 6, 8, 6, u, v, 1, 2, 1, 3, 1, v, 4, 5, with a cost of 15. (b) shows a Longest-Path Walk (LP-Walk): $2, 1, 3, 1, v, 4, 5, 4, v, u, 6, 7, 6, 8$, with a cost of 14. ■ and ◄– indicate the first and last node in a walk, respectively.

13. **RETURN** $W_1, W_2, ..., W_a$, $\sum_{1 \le j \in a} c(W_j)$.

Time Complexity. Using heap data structure, it takes $O(|E|\log q)$ to find the q smallest edges. Line 3-10 takes $O(q)$. Therefore the time complexity of Algorithm 1 is $O(|E|\log q)$.

Theorem 2: Algorithm 1 is optimal in linear topologies.
Proof: By way of contradiction, assume that Algorithm 1 is not optimal and another algorithm, referred to as O, is optimal. In O, since q nodes need to be visited and the topology is linear, q edges must be selected. Denote these q edges selected in O as e_1^o, e_2^o, ..., e_q^o. Since the q edges selected in Algorithm 1 e_1, e_2, ..., e_q are the q smallest-weight edges, it must be that $\sum_{i=1}^{q} e_i \le \sum_{i=1}^{q} e_i^o$, contradicting that O is optimal. ∎

C. Approximation Algorithm for General Graph Topologies.

We first give a few definitions.

Definition 1: (Edge-Induced Subgraph.) An edge-induced subgraph of $G(V, E)$, denoted as $G[E'](V', E')$, is a subgraph of $G(V, E)$ that has edge set $E' \subseteq E$, and for all $(u, v) \in E$, $u, v \in V'$ iff $(u, v) \in E'$. □

Definition 2: (Connected Components of An Edge-Induced Subgraph.) The set of connected components of an edge-induced subgraph $G[E']$, denoted as $C(G[E'])$, is a set of connected subgraphs of $G[E']$. The j^{th} connected component is denoted as C_j, $1 \le j \le |C(G[E'])|$. □

Definition 3: (Cycleless Edges.) An edge $e \in E$ is cycleless w.r.t. $E' \subseteq E$, if $e \notin E'$ and $E' \cup \{e\}$ does not induce a new cycle with connected components of $G[E']$. □

Definition 4: (Binary Walk (B-Walk) In A Tree.) Given a maximum-weight edge (u, v) in tree T (choose one randomly if there are multiple), T can be divided into edge (u, v) and two subtrees T_u and T_v, rooted at u and v respectively. The B-walk of T starts from u and visits all the nodes in T_u in a sequence following depth-first-search (DFS) and comes back, then visits v, from where it visits all the nodes in T_v following DFS. It stops as soon as all the nodes in T are visited. □

Fig. 4(a) shows a tree T with $w(u, v) = 2$ and weights of other edges being 1. It also shows one B-walk, which has a cost of 15. In any B-walk of T, (u, v) is traversed once, each

edge in T_u is traversed twice, and each edge in T_v is traversed once or twice.

Lemma 1: Let $W_B(T) = \{u_1 = u, u_2, ..., u_n\}$ be a B-walk in T. Let $c(T) = \sum_{e \in T} w(e)$. Then $c(W_B(T)) \leq (2 - \frac{1}{|T|}) \times c(T)$, where $|T|$ is the number of edges in T.

Proof: Since (u, v) is the edge in T with maximum weight, $w(u, v) \geq \frac{1}{|T|} \times c(T)$. In $W_B(T)$, since (u, v) is traversed exactly once and other edges are traversed *at most* twice, $c(W_B(T)) \leq \big(2 \times c(T) - w(u, v)\big)$. Therefore $c(W_B(T)) \leq \big(2 \times c(T) - \frac{1}{|T|} \times c(T)\big) = (2 - \frac{1}{|T|}) \times c(T)$. ∎

Approximation Algorithm for MTSW. Next we present an polynomial approximation algorithm, which yields a total cost of the walks that is at most $(2 - \frac{1}{q})$ times of the optimal cost. Algorithm 2 works as follows. Line 1 sorts all the edges in E into nondecreasing order of their weights. Line 2 initializes a set E_q to be an empty set and creates $|V|$ trees, each containing one node. The while loop in lines 3-9 checks if each edge in E (in nondecreasing order of their weights) is cycleless w.r.t. E_q. If yes, add it into E_q. This continues until q edges are added into E_q. It then obtains $G[E_q]$, the subgraph of G induced by E_q (line 10). Since there is no cycles introduced, each connected component of $G[E_q]$ is either linear or a tree. If it is linear, it starts from one end and visits the rest nodes exactly once; if it is a tree, it does a B-walk (lines 11-15).

Algorithm 2: Approximation Algorithm for MTSW.
Input: $G(V, E)$ and number of nodes to visit q;
Output: set of a walks: $W_1, W_2, ..., W_a$, and $\sum_{1 \leq j \in a} c(W_j)$;
1. Let $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|E|})$;
2. $E_q = \phi$ (empty set), $i = j = k = 1$;
3. **while** $(k \leq q)$
4. **if** (e_i is a cycleless edge w.r.t. E_q)
5 $E_q = E_q \cup \{e_i\}$;
6. $k++$;
7. **end if;**
8. $i++$;
9. **end while;**
10. Let $|C(G[E_q])| = a$; /*a connected components*/
11. **for** $(1 \leq j \leq a)$
12. **if** (C_j is linear) Starts from one end of C_j and visits the rest nodes in C_j once;
13. **if** (C_j is a tree) Do a B-walk in C_j;
14. Let the resulted walk (or path) be W_j;
15. **end for;**
16. **RETURN** $W_1, W_2, ..., W_a$, and $\sum_{1 \leq j \in a} c(W_j)$.

Discussions. Using disjoint-set data structure [8], Algorithm 2 takes $O(|E|\log|E|)$. It works alike the well-known Kruskal's minimum spanning tree (MST) algorithm [8], except that instead of finding $|V|-1$ edges to connect all the nodes in V, it only finds $q \leq |V|-1$ edges, to "connect" *some* nodes in V. Therefore Algorithm 2 generalizes Kruskal's MST algorithm in which $q = |V| - 1$. Consequently, a MST is a special case of $G[E_q]$, a *minimum q-edge forest* formally defined below. That is, a MST is a minimum q-edge forest with $q = |V| - 1$.

Definition 5: (**Forest, q-Edge Forest, and Minimum q-Edge Forest**) A forest F of G is a subgraph of G that is acyclic (and possibly disconnected). A q-edge forest F_q is a forest with q edges. The cost of F_q, denoted as $c(F_q)$, is the sum of weights of all edges in F_q, $c(F_q) = \sum_{e \in F_q} w_e$. Let $\mathcal{F}_q$ be set of all q-edge forests of G. A q-edge forest F_q^m is minimum iff $c(F_q^m) \leq c(F_q), \forall F_q \in \mathcal{F}_q$. □

Next we prove that $G[E_q]$ is a minimum q-edge forest. Therefore $G[E_q]$ generalizes MST, and Algorithm 2 generalizes Kruskal's MST algorithm.

Lemma 2: $c(G[E_q]) \leq c(F_q), \forall F_q \in \mathcal{F}_q$.

Proof: Let $E = \{e_1, e_2, ..., e_{|E|}\}$, with $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|E|})$. Let $E_q = \{e_1^g, e_2^g, ..., e_q^g\}$, with $w(e_1^g) \leq w(e_2^g) \leq ... \leq w(e_q^g)$. By way of contradiction, assume that another q-edge forest, O_q, is a minimum q-edge forest with smaller cost than that of $G[E_q]$. Let $O_q = \{e_1^o, e_2^o, ..., e_q^o\}$ with $w(e_1^o) \leq w(e_2^o) \leq ... \leq w(e_q^o)$. Assume that $e_l^g \in E_q$ and $e_l^o \in O_q$ are the first pair of edges that differ in E_q and O_q: $e_l^g \neq e_l^o$ and $e_i^g = e_i^o$, $\forall\ 1 \leq i \leq l-1$. According to Algorithm 2, $w(e_l^g) \leq w(e_l^o)$. Now consider subgraph $O_q \cup \{e_l^g\}$.

Case 1: $O_q \cup \{e_l^g\}$ is a forest. Then $c(O_q \cup \{e_l^g\} - \{e_l^o\}) \leq c(O_q)$, contradicting that O_q is a minimum q-edge forest.

Case 2: $O_q \cup \{e_l^g\}$ is not a forest, i.e., there is a cycle in it. e_l^g must be in this cycle since there is no cycle in O_q. Besides, among all the edges in this cycle that is not e_l^g, at least one of them is not in $\{e_1^g, e_2^g, ..., e_{l-1}^g\}$; otherwise there will not be any cycle. Denote this edge as e'. Let e_l^g be the n^{th} edge in $E = \{e_1, e_2, ..., e_{|E|}\}$, that is $e_l^g = e_n, 1 \leq n \leq |E|$.

Case 2.1: $e' \in \{e_1, e_2, ..., e_{n-1}\}$. Thus $w(e') \leq w(e_{n-1}) \leq w(e_n) = w(e_l^g) \leq w(e_l^o)$, contradicting that e_l^g and e_l^o are the first pair of edges that differ in E_q and O_q.

Case 2.2: $e' \in \{e_{n+1}, e_{n+2}, ..., e_{|E|}\}$. Thus $w(e') \geq w(e_{n+1}) \geq w(e_n) = w(e_l^g)$. $c(O_q \cup \{e_l^g\} - \{e'\}) \leq c(O_q)$, contradicting that O_q is a minimum q-edge forest.

Reaching contradiction in all the cases, it concludes that $c(G[E_q]) \leq c(F_q), \forall F_q \in \mathcal{F}_q$. ∎

Let O be an optimal algorithm of MTSW with minimum cost of $\mathcal{O}$. Next we show $c(G[E_q])$ is a lower bound of $\mathcal{O}$.

Lemma 3: $c(G[E_q]) \leq \mathcal{O}$.

Proof: Without loss of generality, assume that all the edges selected in O induces λ connected components, denoted as O_j $(1 \leq j \leq \lambda)$. Assume that there are l_j nodes in O_j, and s_j $(l_j > s_j \geq 1)$ of them are starting nodes (therefore there are s_j walks in O_j, denoted as W_j^o, visiting altogether $l_j - s_j$ nodes). Denote the s_j walks in O_j as W_j^o and let $c(W_j^o)$ be its cost. We have $\sum_{j=1}^{\lambda} c(W_j^o) = \mathcal{O}$.

Let $c(O_j) = \sum_{e \in O_j} w(e)$. Denote any spanning tree of O_j as T_j^o, and let $c(T_j^o) = \sum_{e \in T_j^o} w(e)$. We have $c(T_j^o) \leq c(O_j) \leq c(W_j^o)$. The first inequality is because all the edges in T_j^o are in O_j (but not vice versa); the second inequality is because each edge in O_j is traversed at least once in O. Therefore $\sum_{j=1}^{\lambda} c(T_j^o) \leq \sum_{j=1}^{\lambda} c(W_j^o) = \mathcal{O}$.

Let $q' = \sum_{j=1}^{\lambda} |T_j^o|$, where $|T_j^o|$ is the number of edges in T_j^o. We have $q' = \sum_{j=1}^{\lambda} (l_j - 1)$. The subgraph induced by all T_j^o $(1 \leq j \leq \lambda)$ is therefore a q'-edge forest. Since all

together q nodes are visited, $\sum_{j=1}^{\lambda}(l_j - s_j) = q$. Since $s_j \geq 1$, we have $q \leq \sum_{j=1}^{\lambda}(l_j - 1) = q'$. Therefore, $c(G[E_q]) \leq c(G[E_{q'}]) \overset{\text{Lemma 2}}{\leq} \sum_{j=1}^{\lambda} c(T_j^o) \leq \mathcal{O}$. ∎

Theorem 3: Algorithm 2 is a $(2 - \frac{1}{q})$-approximation algorithm for MTSW under general graph topologies.

Proof: In Algorithm 2, each of the a connected components C_j $(1 \leq j \leq a)$ is either linear or a tree. Let q_j and $c(C_j)$ denote the number of edges in C_j and the sum of weights of edges in C_j, respectively. We have $q = \sum_{j=1}^{a} q_j$ and $c(G[E_q]) = \sum_{j=1}^{a} c(C_j)$. Let W_j be a B-DFS walk of C_j. Then $\sum_{j=1}^{a} c(W_j) \overset{\text{Lemma 1}}{\leq} \sum_{j=1}^{a}\left((2 - \frac{1}{q_j}) \times c(C_j)\right) < \sum_{j=1}^{a}\left((2 - \frac{1}{q}) \times c(C_j)\right) = (2 - \frac{1}{q}) \times c(G[E_q]) \overset{\text{Lemma 3}}{\leq} (2 - \frac{1}{q}) \times \mathcal{O}$. ∎

Smaller-Tree-First-Walk (STF-Walk). When a B-Walk traverses T_u first and then T_v, each edge in T_u is traversed twice while each edge in T_v is traversed once or twice. A simple improvement is to traverse, between T_u and T_v, the one with smaller cost first. We refer to this as *smaller-tree-first-walk (STF-walk)*. The walk in Fig. 4(a) is indeed a STF-walk.

A Heuristic Algorithm. Next we present a heuristic algorithm to further improve the performance upon Algorithm 2. It differs with Algorithm 2 only in line 13: Instead of a B-walk along each tree C_j $(1 \leq j \leq a)$, it follows a *longest-path-based walk* defined below.

Definition 6: (**Longest-Path Walk (LP-Walk) In A Tree.**) Let $P = \{v_1, v_2, ..., v_n\}$ be the longest path in tree T (choose one randomly if there are multiple). A LP-walk starts from v_1, visiting all the nodes in T in a sequence following DFS, and ends at v_n, such that every edge in P is traversed once. □

Finding longest path in a tree is to find the shortest path among all pair of leaf nodes and choose the longest one. It takes $O(|V|^3)$. LP-walk is based on the observation that when more edges are traversed only once, the cost of a walk can be further reduced. Fig 4(b) shows such a LP-walk, which has a cost of 14. Because the maximum-weight edge (u, v) is not necessarily on the longest path P, we are not able to obtain performance guarantee for LP-walk. However, we show empirically in Section VI that it outperforms Algorithm 2 by $15\% - 30\%$, in terms of energy consumption, under different network parameters.

IV. Equivalency Between MTSW and DAO

Now we transform the original sensor network $G(V, E)$ into an *aggregation network* $G'(V', E')$, and show that solving DAO in G is equivalent to solving MTSW in G'.

Definition 7: (**Aggregation Network** $G'(V', E')$.) V' is the set of p data nodes in V, i.e. $V' = V_d$. For any two data nodes $u, v \in V'$, there exists an edge $(u, v) \in E'$ if and only if all the shortest paths between u and v in G do not contain any other data nodes except u and v. For each edge $(u, v) \in E'$, its weight $w(u, v)$ is the cost of the shortest path between u and v in G. □

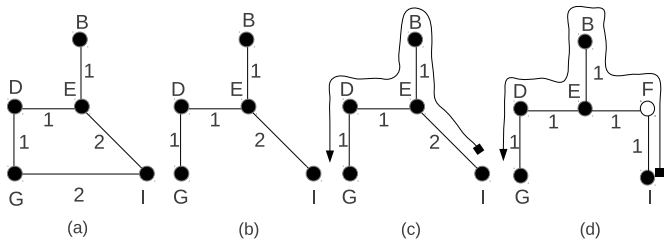


Fig. 5. (a) Aggregation network G' of sensor network G in Fig. 2. (b) 4-edge forest F_q. (c) B-walk on F_q. (d) Aggregation walk with total aggregation cost of 6. The numbers on edges are their weights.

Fig. 5(a) shows the aggregation network G' of sensor network G in Fig. 2. Fig. 5(b) shows a 4-edge forest F_q of G' obtained from Algorithm 2. Fig. 5(c) shows the B-walk on F_q. Finally, we obtain the aggregation walk in G, shown in Fig. 5(d), by replacing each edge (u, v) in F_q with a shortest path between u and v in G (choose one randomly if there are multiple). The total aggregation cost following this walk is 6, one more than the optimal cost shown in Example 1. The B-walk in this example happens to be a LP-walk.

The aggregation network is a novel graph structure that is different from the metric completion of a graph. Metric completion of G is a complete graph wherein the length of edge between every pair of nodes in the graph equals to the length of the shortest path between them in G. The aggregation network resembles the metric completion of graph in the definition of the edge weight or length. However, aggregation network of G is not necessarily a complete graph, due to the intrinsic multi-hop nature of a wireless sensor network. Next we show that an optimal solution of MTSW in G' yields an optimal solution of DAO in G.

Theorem 4: DAO in G is equivalent to MTSW in G'.

Proof: First, we argue that any storage node in G does not appear in G'. For any two data nodes X and Y, as DAO concerns with visiting data nodes following shortest paths, all the storage nodes on the shortest paths between X and Y in G do not appear in G', as long as the total aggregation cost (i.e., the energy consumption of sending data from X to Y) is accurately captured. This is accomplished by the aggregation network G' defined above since the weight of the edge (X, Y) in G' represents the energy consumption along the shortest paths between X and Y.

Second, we argue that if all the shortest paths between two data nodes X and Y in G do not contain any other data nodes, then in G' they can be replaced by one single edge (X, Y), whose weight is the cost of any of such shortest paths. This is due to the same reason that storage nodes are not included in G' as long as the energy consumption information is accurately captured. For example, in Fig. 2, both of the two shortest paths between data nodes E and I do not contain another data node, therefore edge (E, I) appears in G', as shown in Fig. 5.

Finally, if there exists multiple shortest paths between two data nodes X and Y in G, some of which having at least another data node as intermediate nodes and some not, edge

(X, Y) is not included in G'. This is because in order to visit as many data nodes (aggregators) as possible while using as least amount of energy as possible, it mandates that DAO takes a shortest path with maximum number of data nodes as intermediate nodes (Otherwise, less number of aggregators are visited with the same amount of energy consumption, which is against the goal of DAO). Therefore, if there is an aggregation route from X to Y, it must go through maximum number of intermediate data nodes (for the purpose of aggregation) along a shortest path between X and Y, resulting in that edge (X, Y) does not appear in G'. For example, in Fig. 2, edge (B, I) does not appear in the aggregation network in Fig. 5, since one of the shortest path between B and I contains the data node E.

Since all the information in G that is used for data aggregation calculation, including data nodes and energy consumption sending data among them, are accurately captured in the aggregation network G', solving MTSW in G' is equivalent to solving DAO in G. ∎

V. Distributed Data Aggregation Algorithm

We design a distributed data aggregation algorithm for the overall storage overflow problem, referred to as *Distributed DAO*. It is based on the classic GHS algorithm, a distributed, asynchronous algorithm by Gallager, and Humblet, and Spira [11] that finds the optimal minimum spanning tree (MST) of a graph.[3] Below we present Distributed DAO and its time and message complexities, and discuss its optimality.

Distributed DAO. It has two stages. First, it needs to find the aggregation network $G'(V', E')$ in a distributed manner. This is equivalent to finding the shortest path between each pair of data nodes in sensor network $G(V, E)$, which can be done by distributed Bellman-Ford algorithm [29]. Second, it finds the minimum q-edge forest in $G'(V', E')$ distributedly. Most steps in second stage are similar with those in original GHS algorithm. We therefore review GHS algorithm first.

The algorithm maintains a spanning forest of trees, each being a sub-tree of the MST. It starts with each node being considered as a fragment, which has level 0 initially. In each "round" of the algorithm, each fragment independently finds its minimum weight outgoing edge (MWOE) and uses this edge to combine with other fragments. Specifically, each fragment has its leader to manage the combining operations, which are either "merge" or "absorb" operations. Absorb operation doesn't change the maximum level among all fragments while merge operation may increase the maximum level by 1. To find the MWOE, the leaders of two fragments, which are adjacent to the edge added immediately in the previous step, send initiate message to the members of the fragment. Upon receipt of the initiate message, each member node finds its outgoing edge and reports it to the leaders. Upon receipt of reports, the leaders select a new leader, the node that is adjacent to the MWOE for the entire fragment, and then begins a new round. During the execution of the algorithm, an edge that becomes part of the MST is a *branch* edge. The only difference between Distributed DAO and GHS is the termination case: for GHS, it is when a fragment is unable to find a MWOE; for Distributed DAO, it is when the number of branch edges reaches q.

Time and Message Complexity. We only give analysis results due to space constraint. For the first stage of Distributed DAO, both its time and message complexities are $O(p|E|)$, where p is the number of data nodes. For the second stage, its time complexity is $O(p \cdot \log p)$ while its message complexity is $O((p + |E'|) \cdot \log p)$. Therefore, the first stage dominates and both the time and message complexities of Distributed DAO are $O(p|E|)$.

Theorem 5: When there is one initiator allowed, Distributed DAO finds an optimal aggregation cost.

Proof: When there is only one initiator in the data aggregation, $q = |V| - 1$. Therefore finding minimum q-edge forest in data aggregation is equivalent to finding a MST in the aggregation network. Consequently, Distributed DAO is equivalent to GHS algorithm. Due to the optimality of finding MST by the GHS algorithm, the optimality of Distributed DAO also sustains. ∎

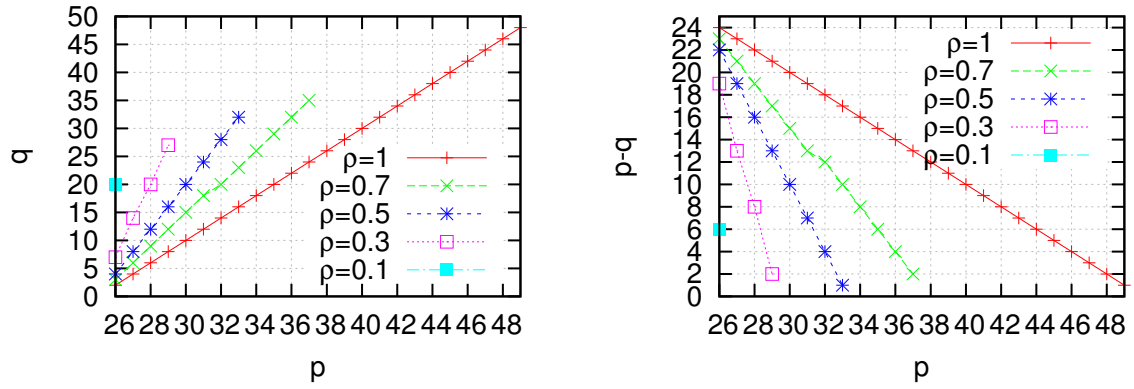
VI. Performance Evaluation

A. Centralized Algorithms.

Our centralized algorithms are implemented in Java. In our experiments, 50 and 100 sensors are uniformly distributed in a region of 1000m × 1000m. Transmission range is 250m. Unless otherwise mentioned, $R = m = 512$KB. We define *correlation coefficient* as $\rho = 1 - r/R$. $\rho = 0$ means no spatial correlation at all, while $\rho = 1$ means perfect correlation (i.e., data at aggregators are duplicate copies of data at initiators thus can be completely removed). In the plots, each data point is an average over 10 runs and the error bars indicate 95% confidence interval, wherever applicable. Since the comparison results are similar for 50 and 100 nodes, we only present the results for 50 nodes, because they can be clearly visualized.

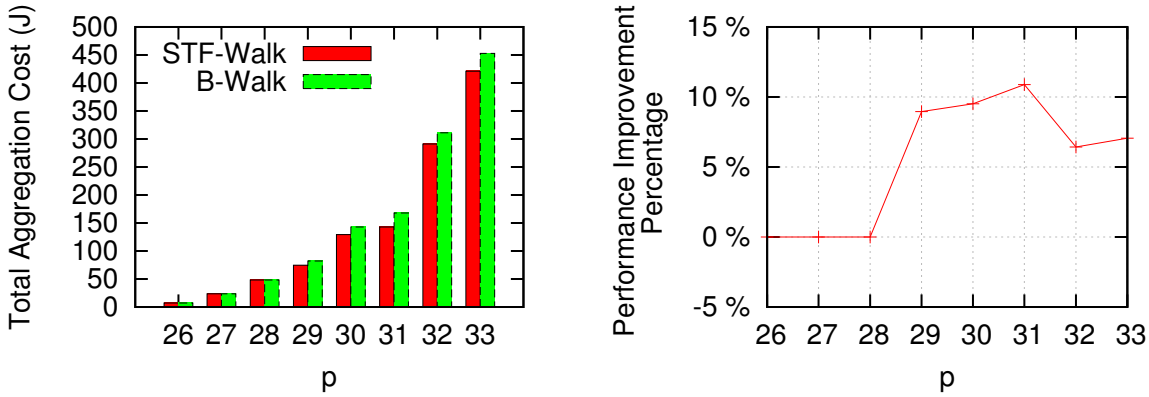
Valid Range of p. Fig. 6(a) shows the valid range of p for different correlation coefficient ρ. When $\rho = 0.1$, the valid range of p is a single value of 26, with corresponding value of q, the number of aggregators, as 20. When increasing ρ, the valid range of p expands, from $26 - 29$ for $\rho = 0.3$, to $26 - 33$ for $\rho = 0.5$, to $26 - 37$ for $\rho = 0.7$, to $26 - 49$ for $\rho = 1$. This is because strong data correlation leads to more data being aggregated, thus allowing more data nodes to exist under overall storage overflow. It also shows that for each ρ, q increases when increasing p. This is because more data nodes means more overflow data and less available storage, therefore more aggregators are needed to achieve enough data size reduction. Finally it shows that for the same p, q decreases when increasing ρ. This is implied by Equation 1, which can be rewritten as: $q = \lceil \frac{p \times (1 + m/R) - |V| \times m/R}{\rho} \rceil$. Fig. 6(b) shows the maximum number of initiators $p - q$ for each valid p value. There are two cases that one initiator is allowed: $\rho = 0.5$ and $p = 33$, and $\rho = 1$ and $p = 49$, while multiple initiators are

[3] There are a few work [7, 18] that construct distributed MST in wireless ad hoc networks with less energy consumption. Their distributed MST is an $O(|V| \log |V|)$-approximation to the optimal MST.



(a) q vs. p. (b) $p - q$ vs. p.

Fig. 6. Valid range of p while varying ρ ($R = m$).



(a) Total aggregation cost. (b) Performance improvement.

Fig. 7. Performance improvement of STF-Walk over B-Walk.

allowed for other cases. We study both cases of single initiator and multiple initiators.

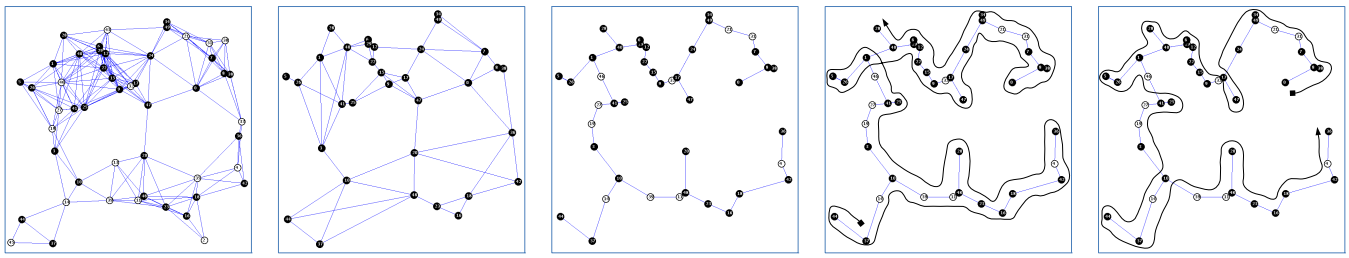
Performance Improvement of STF-Walk Over B-Walk. We first study the performance improvement of STF-Walk over B-Walk. We choose $\rho = 0.5$, which is a representative correlation coefficient, and vary p from 26 to 33. Fig. 7(a) shows that when p is 26, 27, or 28, both STF-Walk and B-Walk yield the same total aggregation costs. This is because when the number of data nodes p is small, the number of aggregators q is small, causing that the connected components of the resulted q-edge forests are all linear. In linear topologies, aggregation takes place by simply traversing from one end of the linear topology to the other end, resulting the same performances for both STF-Walk and B-Walk. However, when p gets larger, STF-Walk yields less cost and performs better than B-Walk does, because STF-Walk always traverses the smaller subtree twice while B-Walk could possibly traverse the bigger subtree twice. Fig. 7(b) shows that the performance improvement of STF-Walk over B-Walk is around $5\% - 10\%$. Therefore, for the rest of the simulations we choose STF-Walk instead of B-Walk, but still refer to it as B-Walk.

Comparing B-Walk with LP-Walk Visually. Before we perform a comprehensive comparison between B-Walk and LP-Walk, we first compare them visually to gain some insights.

Single Initiator Case. We consider $\rho = 0.5$ and $p = 33$, which has 32 aggregators and one initiator. Fig. 8(a) and (b) show such a sensor network and its corresponding aggregation network, respectively. Fig. 8(c) shows the corresponding 32-edge forest.[4] Fig. 8(d) and (e) show the aggregation walks from B-Walk and LP-Walk, respectively. B-Walk visits 32 edges twice, resulting in a total aggregation cost of $381.2J$; while LP-Walk only visits 12 edges twice, with a total cost of $290.6J$, a 23.8% of improvement upon B-Walk.

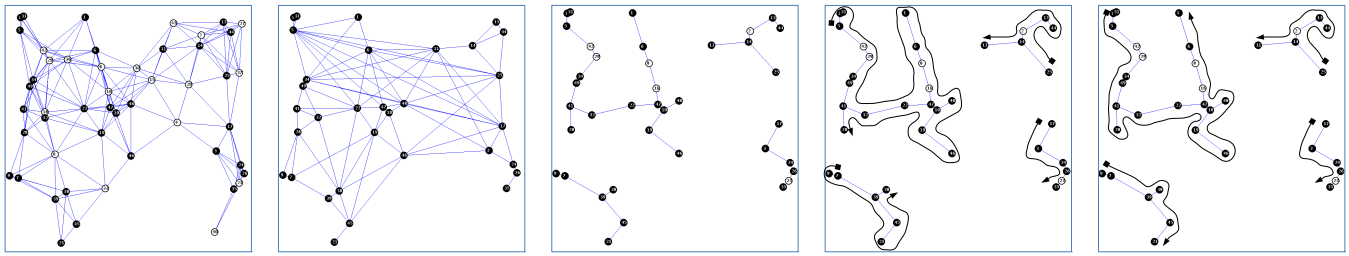
Multiple Initiators Case. We consider $\rho = 0.5$ and $p = 32$, which has 28 aggregators and allows at most 4 initiators. Fig. 9(a) and (b) show such a sensor network and its corresponding aggregation network. Fig. 9(c) shows the 28-edge forest of the aggregation network. It consists of four "clusters", each of which is visited by one initiator. However, Fig. 9(d)

[4]Here, each edge in the 32-edge forest is replaced by a shortest path of storage nodes, if there are any. Therefore there are more than 32 edges in this forest. The multiple initiators case below is similar.



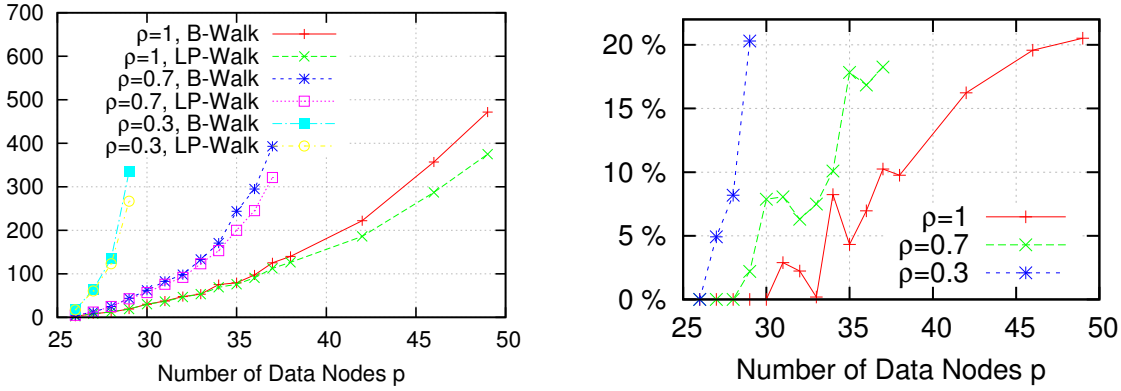
(a) Sensor network of 50 nodes. (b) Aggregation network. (c) 32-edge forest. (d) B-Walk (381.2J). (e) LP-Walk (290.6J).

Fig. 8. Visually comparing B-Walk with LP-Walk with one initiator. Black nodes are data nodes and white nodes are storage nodes, with node ID shown inside. Here, $\rho = 0.5$, $p = 33$, and $q = 32$. ■ and ◄– indicate the first and last node in a walk, respectively.



(a) Sensor network of 50 nodes. (b) Aggregation network. (c) 28-edge forest. (d) B-Walk (cost 255.9J). (e) LP-Walk (cost 203.0J).

Fig. 9. Visually comparing B-Walk with LP-Walk with 4 initiators. Black nodes are data nodes and white nodes are storage nodes, with node ID shown inside. Here, $\rho = 0.5$, $p = 32$, and $q = 28$. ■ and ◄– indicate the first and last node in a walk, respectively.

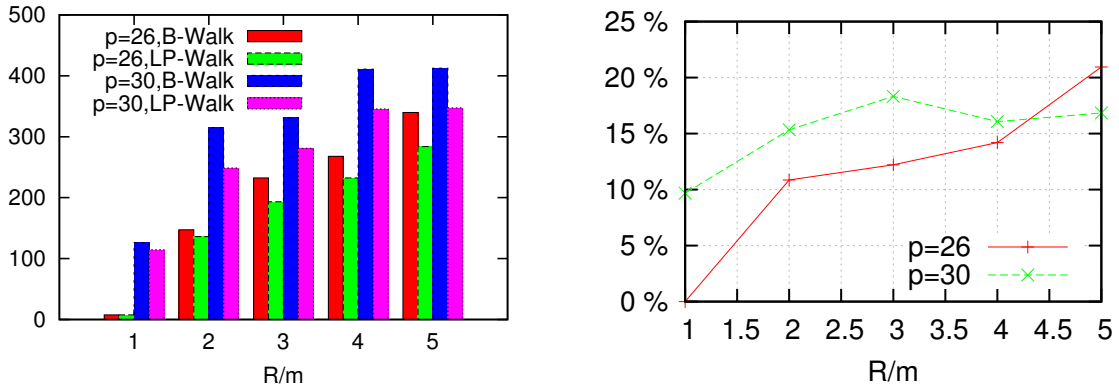


(a) Total aggregation cost (J). (b) Performance improvement.

Fig. 10. Comparing B-Walk with LP-Walk by varying p and ρ.

and (e) show that how B-Walk and LP-Walk visit each cluster differently. It shows that B-Walk traverses 19 edges twice, resulting in a total aggregation cost of 255.9J, while LP-Walk traverses 9 edges twice, with a total aggregation cost of 203.0J, a 20.7% of improvement. Compared to Fig. 8, this shows that when increasing number of initiators, the performance difference between B-Walk and LP-Walk gets smaller. In particular, Fig. 9(d) and (e) show that they find exactly the same aggregation walks for two smaller trees. This is because with more initiators allowed, the resulted q-edge forest consists of more trees with smaller sizes, each with a "short" longest path. By traversing edges on such short longest paths once, LP-Walk does not save as much at it does compared to traversing a big tree with much longer longest path. Finally, compared to single initiator case, both B-Walk and LP-Walk incur less energy cost, because more initiators are utilized to find more cost-effective aggregation walks.

Comparing B-Walk with LP-Walk by Varying p and ρ. Next we compare B-Walk and LP-Walk while considering the whole ranges of $p \in [26, 49]$ and $\rho = 0.1, 0.3, 0.5, 0.7, 1.0$. Fig. 10(a) shows that for each ρ, with the increase of p, the total aggregation costs of both B-Walk and LP-Walk increase. However, LP-Walk constantly performs better than B-Walk.



(a) Total aggregation cost (J).

(b) Performance improvement.

Fig. 11. Comparing B-Walk with LP-Walk by varying R/m.

It also shows that for the same p, with the increase of ρ, the aggregation costs for both B-Walk and LP-Walk decrease. This is because more correlation means that less number of aggregators are visited, thus reducing aggregation costs.

Fig. 10(b) shows the performance improvement percentage of LP-Walk over B-Walk is generally $10\% - 20\%$. Combining the $5\% - 10\%$ performance improvement of STF-Walk over B-Walk, the performance improvement of LP-Walk over B-Walk is therefore around $15\% - 30\%$. Furthermore, we observe the smaller the ρ, the larger the performance improvement percentage is. For example, when $p = 26$ (the only valid value for $\rho = 0.1$), the performance improvement percentage for $\rho = 0.1$ is 14% while zero for $\rho = 0.3, 0.5, 0.7, 1.0$. When $\rho = 0.5$, in its valid p range $(26 - 33)$, it almost always has a larger performance improvement percentage compared to $\rho = 0.7, 1$. When less data correlation exists, more aggregators are visited, making the sizes of the resulted q-edge forest as well as its constituent trees larger. By traversing the longest paths of larger trees once, LP-Walk can thus save more aggregation cost compared to traversing smaller trees.

Comparing B-Walk with LP-Walk by Varying R/m. We compare B-Walk with LP-Walk on different R/m. When increasing R/m, the overall storage overflow situation gets more challenging since there are relatively more overflow data compared to available storage spaces. We choose $\rho = 0.5$ and vary R/m from 1 to 5, under which the common valid range of p is $[26, 30]$. Therefore we pick $p = 26$ and $p = 30$ for comparison. Fig. 11(a) shows again that LP-Walk yields less total aggregation cost under different R/m. Fig. 11(b) further shows that the performance improvement percentage of LP-Walk upon B-Walk generally increases when increasing R/m. This indicates that LP-Walk performs even better in more challenging overall storage overflow scenarios. The observation is that when increasing R/m, the resulted q-edge forests get larger. This favors LP-Walk, which travels large amount of edges only once.

B. Distributed Algorithm.

Finally, we evaluate the performance of the Distributed DAO. Distributed DAO is implemented in Python and is based on DistAlgo [25], a high-level language for clear description of distributed algorithms. 100 nodes are randomly placed in a 2000m $\times$ 2000m region. The transmission range is 250m and $m = R = 512$KB. We set the size of each overhead message as 20B. Under above parameters, $\rho = 0.6$ and $p = 71$ allow for one initiator case. Table I compares the aggregation costs on the q-edge forests resulted from the Distributed DAO as well as the centralized approximation algorithm (Algorithm 2), while varying number of data nodes p in its valid range [55,71]. It shows that even though centralized algorithm performs much better than Distributed DAO when p is small, Distributed DAO performs more and more closer to the centralized algorithm when increasing p. When it gets to the case of one initiator ($p = 71$), Distributed DAO yields the same aggregation cost as the centralized algorithm does, since they both result in the minimum q-edge forest, which is a minimum spanning tree. Fig. 12 compares the total energy cost, which for Distributed DAO, also includes the energy consumptions for the aggregation network construction and overhead messages. While centralized algorithm obviously costs less energy than Distributed DAO, the energy consumptions of them are generally very comparable. This shows that our distributed data aggregation algorithm is energy-efficient.

VII. Related Work

MSTW is different from well-known multiple traveling salesman problem (mTSP) [5] and vehicle routing problem (VRP) [39] studied in theory community. In mTSP, a group of traveling salesmen are given, and it needs to decide a tour for each salesman such that the total tour cost is minimized and that each city is visited exactly once. MSTW, however, needs to first decide how many salesmen can be dispatched (and from which cities), then to find the tour for each. Besides, not necessarily all the nodes will be visited in MSTW. In VRP, the set of vehicle nodes and the set of customer nodes are usually

p	55	60	65	70	71
q	17	34	50	67	70
Number of Initiators	38	26	15	3	1
Centralized (J)	78.79	251.76	494.12	787.07	876.29
Distributed (J)	209.52	479.12	680.93	827.76	876.29

TABLE I
AGGREGATION COSTS IN CENTRALIZED AND DISTRIBUTED ALGORITHMS.

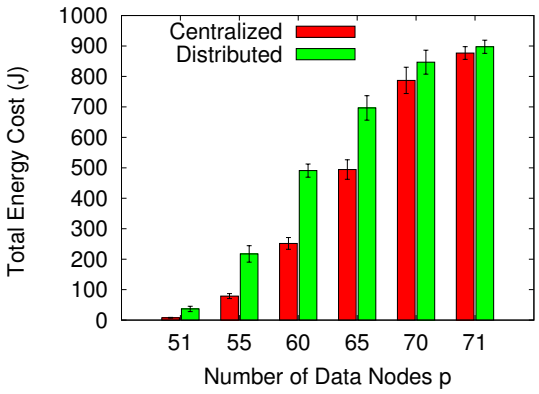


Fig. 12. Total energy consumption in centralized and distributed algorithms.

disjoint. There is no such distinction in MTSW – each node can either dispatch a salesman or be visited.

In sensor network community, there are active research that focused on disconnection-tolerant operations in the absence of the base station. Some system research were conducted to design cooperative distributed storage systems and to improve the utilization of the networks data storage capacity [27, 28]. Other research instead took an algorithmic approach by focusing on the optimality of the solutions [14, 37, 44]. However, all above works assume that there is enough storage space available to store the overflow data, thus not addressing the overall storage overflow problem.

There is vast amount of literature of data aggregation in sensor networks [3, 19, 23, 26, 36, 38, 43]. Here we only review the most recent and most related works. Tree-based routing structures were often proposed to either maximize the network lifetime (the time until the first node depletes its energy) [26, 43], or minimize the total energy consumption or communication cost [19, 23], or reduce the delay of data gathering [3].

Data aggregation in DAO, however, significantly differs from existing data aggregation techniques both goals and techniques. First, existing data aggregation is to reduce number of transmissions by combining data from different sensors en route to base station, thus saving energy. The goal of data aggregation in DAO, however, is to aggregate the overflow data so that they can fit into storage available in the network, thus preventing data loss caused by overall storage overflow. Second, most of the existing data aggregation techniques wherein the underlying routing structures are trees rooted at the base station covering all sensor nodes. In DAO, however, since the base station is not available and data must be stored inside the network, tree-bases routing structure is no longer suitable. Instead, DAO introduces a novel routing structure called *minimum q-edge forest*, which is the key that enable an approximation algorithm with constant performance ratio, and a new variation of the traveling salesman problem [21].

Some other works were based on non-tree routing structures, using mobile base stations to collect aggregated data in order to maximize the network lifetime [36, 38]. In contrast, we address a very different problem for sensor networks: before the mobile base stations or data mules become available, some sensor nodes already deplete their storage. It therefore calls for new data aggregation techniques that not only reduce the size of sensory data while sacrificing no information loss, but also guarantee that aggregated data can be stored inside the network. To the best of our knowledge, the overall storage overflow problem has not been addressed by any of the existing data aggregation research.

VIII. Conclusion and Future Work

In this paper we tackled the overall storage overflow problem (DAO) in base station-less sensor networks, wherein the generated sensory data overflows the storage capacity available in the entire network. To solve the DAO, we designed a suite of energy-efficient optimal, approximation, heuristic, and distributed data aggregation algorithms. The novelties of our work are two fold. First, underlying the DAO is a new multiple traveling salesman walks problem (MTSW), which has not been studied before. We proved its NP-hardness and designed a constant ratio approximation algorithm for it. Second, the minimum q-edge forest uncovered in this paper generalizes minimum spanning tree, and the approximation algorithm designed generalizes the well-known Kruskal's algorithm. Because of these theoretical significance, we believe that the techniques proposed in this paper could potentially be of interest to broader applications.

As future work, we will first extend the uniform data size reduction by considering that different nodes could have different correlation coefficients. Second, we will design distributed algorithms by considering that overflow data can be generated dynamically, and that some nodes can deplete their battery power, as well as packet losses and retransmission. Third, currently data aggregation and data offloading are treated as two separate stages – we will consider integrating these two stages and explore a more unified energy-efficient solution for

the overall storage overflow problem. For example, in Fig. 2, there are two optimal data aggregation solutions: B is the initiator and its aggregation walk is: B, E, D, G, H, I; or I is the initiator and its aggregation walk is: I, H, G, D, E, B. However, the former achieves optimal for the whole problem while the latter not. As an ongoing effort [2], we have designed two energy-efficient data replication algorithms to integrate data aggregation and data offloading. However, they are heuristic algorithms that do not have any performance guarantees. As future work, we would like to integrate these two stages together to explore a more energy-efficient solution for the overall storage overflow problem, while achieving some performance guarantee.

ACKNOWLEDGMENT

This work was supported in part by NSF Grant CNS-1419952.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] B. Alhakami, B. Tang, J. Han, and M. Beheshti. Dao-r: Integrating data aggregation and offloading in sensor networks via data replication. In *Proc. of the IEEE Global Communications Conference (GLOBECOM 2015)*.

[3] B. Alinia, M. Hajiesmaili, and A. Khonsari. On the construction of maximum-quality aggregation trees in deadline-constrained wsns. In *Proceedings of the IEEE Infocom*, 2015.

[4] S. Basagni, L. Boloni, P. Gjanci, C. Petrioli, C. A. Phillips, and Danila Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *Proc. of Infocom 2014*.

[5] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Elsevier Omega*, 34:209–219, 2006.

[6] C. Busch, M. Magdon-Ismail, F. Sivrikaya, and B. Yener. Contention-free mac protocols for wireless sensor networks. In *Proc. of DISC*, 2004.

[7] Y. Choi, M. Khan, V. A. Kumar, and G. Pandurangan. Energy-optimal distributed algorithms for minimum spanning trees. *IEEE Journal on Selected Areas in Communications*, 27(7):1297–1304, 2009.

[8] T. Corman, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[9] M. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[10] R. Cristescu, B. Beferull-Lozano, M. Vetterli, and R. Wattenhofer. Network correlated data gathering with explicit communication: Np-completeness and algorithms. *IEEE/ACM Transactions on Networking*, 14:41–54, 2006.

[11] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.

[12] J. Heidemann, M. Stojanovic, and M. Zorzi. Underwater sensor networks: applications, advances and challenges. *Phil. Trans. R. Soc. A*, 370:158 – 175, 2012.

[13] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000*.

[14] X. Hou, Z. Sumpter, L. Burson, X. Xue, and B. Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of MASS 2012*, pages 448–452.

[15] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *MONET*, 11(3):327–339, 2006.

[16] I. Jawhar, N. Mohamed, and D. P. Agrawal. Linear wireless sensor networks: Classification and applications. *Journal of Network and Computer Applications*, 34:1671–1682, 2011.

[17] J. Jeong, X. Jiang, and D. Culler. Design and analysis of micro-solar power systems for wireless sensor networks. In *Proc. of INSS 2008*.

[18] M. Khan. *Distributed Approximation Algorithms for Minimum Spanning Trees and Other Related Problems with Applications to Wireless Ad Hoc Networks*. PhD thesis, 2007.

[19] T. Kuo and M. Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. In *Proc. of INFOCOM 2012*.

[20] F. Lam and A. Newman. Traveling salesman path problems. *Mathematical Programming*, 113:39 – 59, 2008.

[21] E. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D. SHmoys (Eds.). *The Traveling Salesman Problem: A Guided Tour of Combinational Optimization.* John Wiley and Sons, 1985.

[22] H. Li, D. Liang, L. Xie, G. Zhang, and K. Ramamritham. Flash-optimized temporal indexing for time-series data storage on sensor platforms. *ACM Trans. Sen. Netw.*, 10(4):1565–1572, 2012.

[23] J. Li, A. Deshpande, and S. Khuller. On computing compression trees for data collection in wireless sensor networks. In *Proc. of INFOCOM 2010*, pages 2115–2123.

[24] K. Li, C. Shen, and G. Chen. Energy-constrained bi-objective data muling in underwater wireless sensor networks. In *Proc. of MASS 2010*, pages 332–341, 2010.

[25] Y. Liu, S. Stoller, B. Lin, and M. Gorbovitski. From clarity to efficiency for distributed algorithms. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 395–410, 2012.

[26] D. Luo, X. Zhu, X. Wu, and G. Chen. Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks. In *Proc. of INFOCOM 2011*, pages 1566 – 1574.

[27] L. Luo, Q. Cao, C. Huang, L. Wang, T. Abdelzaher, and J. Stankovic. Design, implementation, and evaluation of enviromic: A storage-centric audio sensor network. *ACM Transactions on Sensor Networks*, 5(3):1–35, 2009.

[28] L. Luo, C. Huang, T. Abdelzaher, and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proc. of INFOCOM 2007*.

[29] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[30] M. Ma and Y. Yang. Data gathering in wireless sensor networks with mobile collectors. In *Proc. of IPDPS 2008*.

[31] K. Martinez, R. Ong, and J.K. Hart. Glacsweb: a sensor network for hostile environments. In *Proc. of SECON 2004*.

[32] Ioannis Mathioudakis, Neil M. White, and Nick R. Harris. Wireless sensor networks: Applications utilizing satellite links. In *Proc. of PIMRC 2007*.

[33] D. Mosse and G. Gadola. Controlling wind harvesting with wireless sensor networks. In *Proc. of IGCC 2012*.

[34] L. Mottola. Programming storage-centric sensor networks with squirrel. In *Proc. of IPSN 2010*, pages 1–12.

[35] S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (discus): design and construction. In *Proc. IEEE Data Compression Conference (DCC)*, 1999.

[36] Y. Shi and Y.T. Hou. Theoretical results on base station movement problem for sensor network. In *Proc. of INFOCOM 2008*.

[37] B. Tang, N. Jaggi, H. Wu, and R. Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2), May 2013.

[38] S. Tang, J. Yuan, X. Li, Y. Liu, G. Chen, M. Gu, J. Zhao, and G. Dai. Dawn: Energy efficient data aggregation in wsn with mobile sinks. In *Proc. of IWQoS 2010*.

[39] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2001.

[40] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of SenSys 2005*.

[41] B. Weiss, , H.L. Truong, W. Schott, A. Munari, C. Lombriser, U. Hunkeler, and P. Chevillat. A power-efficient wireless sensor network for continuously monitoring seismic vibrations. In *Proc. of SECON 2011*.

[42] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of OSDI 2006*.

[43] Y. Wu, S. Fahmy, and N. B. Shroff. On the construction of a maximum-lifetime data gathering tree in sensor networks: Np-completeness and approximation algorithms. In *Proc. of INFOCOM 2008*.

[44] X. Xue, X. Hou, B. Tang, and R. Bagai. Data preservation in intermittently connected sensor networks with data priorities. In *Proc. of SECON 2013*, pages 65–73.