Data Resilience Via Data Aggregation: Overcoming Overall Storage Overflow in Sensor Networks

***Abstract*—Data resilience refers to the ability of long-term viability and availability of data despite insufficiencies of (or disruptions to) the physical infrastructure that stores the data. In this paper, we identify, formulate, and solve a data resilience problem that uniquely arises from sensor networks operating in an inaccessible or inhospitable region, or under extreme weather. In these challenging environments, it is not feasible to install a long-term base station in the field. Data generated must be stored inside the network for some period of time before uploading opportunities become available. Consequently, the collected data could soon overflow the storage capacity available in the entire network, making discarding valuable data inevitable. We refer to this data disruption as *overall storage overflow* in sensor networks. To overcome this obstacle, we propose *data resilience via data aggregation problem (DRA)*, which employs data aggregation techniques to reduce the overflow data size so that they can fit into the storage in the network. The goal of DRA is to minimize the total energy consumption during this process while preserving as much information as possible. Our work is the first one to employ data aggregation to achieve data resilience against overall storage overflow problem in sensor networks. We show that solving DRA is equivalent to solving a *multiple traveling salesman walk problem (MTSW)* in an appropriately transformed graph of the sensor network. 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 number of nodes to visit. We design a heuristic algorithm to further improve upon the approximation algorithm, and empirically show that it constantly outperforms the approximation algorithm by $20\% - 30\%$, in terms of energy consumption, under different network parameters.**

Keywords – Data Resilience, Data Aggregation, Overall Storage Overflow, Sensor Networks, Energy-Efficiency

I. Introduction

Background and Motivation. *Data resilience* refers to the ability of a network storing the data to recover quickly and to continue maintaining availability of data despite of any disruptions (such as equipment failure, power outage, or even malicious attack). Due to resource constraints of sensor networks such as unreplenishable battery power and limited storage capacity of sensor nodes [10, 11], link unreliability and scarce bandwidth of wireless medium [45], and the inhospitable and harsh environments in which they are deployed [5], sensor nodes are often prone to failure and vulnerable of data loss. Therefore, how to ensure that data collected at sensor nodes reach the base station reliably despite all the vulnerabilities has been an active research topic since the inception of sensor network research. This line of research are usually named under the umbrella of data resilience [10, 11], data persistence [2, 24], or reliable data transmission [9]. We use data resilience throughout the paper.

Meanwhile, 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, and to solve some of the most fundamental problems facing human beings. These fundamental problems include disaster warning, climate change, and renewable energy. The emerging sensor networks designed for those scientific applications include seismic sensor networks [39], underwater or ocean sensor networks [17], wind and solar harvesting [16, 31], and volcano eruption monitoring and glacial melting monitoring [29, 40]. 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 without much human intervention. Consequently, it is not practical to deploy data-collecting base stations with power outlets in or near such inaccessible sensor fields. Thereupon sensory data generated in such environments have to be stored inside the network for some period of time and then being collected by periodic visits of robots or data mules [15], or by low rate satellite link [30]. Due to the lack of human intervention and the inadequacy of maintenance in the inhospitable environments, such sensor networks must operate much more resiliently than the traditional sensor networks (with base stations and in friendly environments).

Data Resilience Against Storage Overflow Disruption. In this paper, we focus on data resilience against sensor storage overflow, wherein storage spaces of some sensor nodes are depleted and therefore it can not store any newly generated data. Storage overflow is a major obstacle existing in above emerging networks, due to the 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 [32] 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. From scientific perspective, data are "first class citizens" because every bit of data could potentially be important for scientists to analyze the physical world. Thus how to resiliently maintain such overflow data inside the network and prevent

data loss becomes a crucial task. Below we outline two levels of data overflow disruptions, and provide corresponding data resilience measures.

- Node Storage Overflow. Some sensor nodes are close to the events of interest and are constantly generating sensory data, depleting their own storages and causing data loss. We refer to such disruption as *node storage overflow*, and the 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*. The resilience measure to avoid such data loss is simple: the overflow data is offloaded to other nodes with available storages (referred to as *storage nodes*).[1] Different data offloading techniques have been proposed with the goals of either minimizing the total energy consumption during data offloading [35], or maximizing the minimum remaining energy of storage nodes to prolong network lifetime[14], or offloading the the most useful information considering data could have different priorities [43].
- Overall Storage Overflow. However, data offloading can not help when the total size of the overflow data is larger than the total size of the available storage in the network. We refer to this disruption as *overall storage overflow* in sensor networks, wherein discarding data becomes inevitable if no actions taken. This is obviously a more severe disruption compared to the storage flow of individual nodes. For the large amount of overflow data generated in the sensor networks, how to guarantee that as much useful information as possible can be maintained while being fitted into the available storage of the network becomes a new challenge. In this paper, we endeavor to answer following question: *For sensor networks operating in challenging environments where base station is absent and human intervention impossible, how to retain all the information notwithstanding the overall storage overflow?*

Data Aggregation. Fortunately, due to spatial correlation that commonly exist among sensory data collected from sensor networks, we can employ data aggregation techniques to reduce data size. We formulate a graph-theoretic problem called *data resiliency via data aggregation (DRA)*, and solve it by designing a suite of new data aggregation algorithms to aggregate overflow data, so that they can fit into the available storage in the network. Our goal is to minimize total energy consumption during data aggregation, since battery power of sensor nodes is still one of the most stringent resources in sensor networks and data aggregation, being wireless communication, costs most of the battery power. After being aggregated to the size accommodable by the available storage capacity, the overflow data can then be offloaded to storage nodes using techniques proposed in [14, 35, 43].

Difference Between Data Aggregation in DRA and Traditional Data Aggregation. Data aggregation in DRA is significantly different from traditional data aggregation in sensor networks [19, 22, 25, 34, 36, 41, 42] in both goals and techniques.

- In traditional sensor networks (with base stations), data aggregation is to combine the data from different sources en route to the base station, by eliminating redundancy and reducing the total number of transmissions, thus saving energy. The goal of data aggregation in DRA, however, is to aggregate the overflow data so that they can fit into available storage, therefore preventing data loss caused by overall storage overflow. 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.
- Unlike most of the existing data aggregation techniques [19, 22, 25, 41, 42] wherein the underlying routing structures are trees rooted at base station covering all sensor nodes, at the core of the DRA is a novel routing structure called *q-edge forest*, which is a graph with q edges and with each connected component a tree. Here q is number of data nodes that aggregate their overflow data. Algorithmically, DRA encapsulates a new graph-theoretic problem called *multiple traveling salesman walks (MTSW)*, which has not been studied before.

Contributions of This Paper. The main contributions of this paper include the following:

1). We identify, formulate, and solve DRA, the data resilience problem via data aggregation, to address overall storage overflow in sensor networks. (Section II

2), We show that DRA in sensor network is equivalent to a new *multiple traveling salesman walk problem (MTSW)* in a graph appropriately transformed from the sensor network graph. (Section IV)

3). We show that the MTSW is NP-hard, by showing that it generalizes traveling salesman walk problem [20], which is NP-hard. (Section III-A)

4). We solve optimally the MTSW in linear topologies, and design a $(2 - \frac{1}{q})$-approximation algorithm for general graph topologies, where q is number of nodes to visit. (Section III-B)

5). We design a heuristic algorithm to further improve upon the approximation algorithm, and empirically shows that it outperforms the approximation algorithm by $10\% - 20\%$, in terms of energy consumption, under different network parameters. (Section V)

II. Data Resilience Via Data Aggregation Problem (DRA)

Problem Statement. In a sensor network field (without base stations), there are two kinds of sensor nodes: **data nodes** (with overflow data and with storage spaces depleted), and **storage nodes** (with available storage spaces), as shown in Figure 1. The total size of the overflow data from data nodes is larger than the total size of the storage space available at storage nodes, causing overall storage overflow. Therefore overflow data needs to be aggregated to the size that can fit into the available storages, before being offloaded to storage

[1] Sensor nodes that generate data but have not depleted their storage are considered as storage nodes, since they can store overflow data from others.

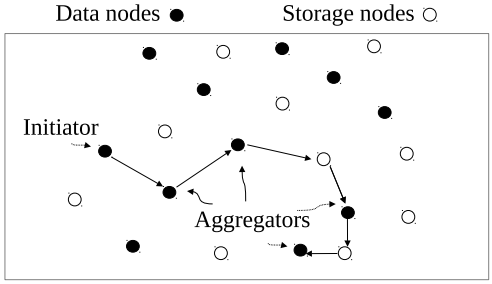


Fig. 1. An illustration of DRA.

nodes. To aggregate data, one (or multiple) data node (called an **initiator**) sends its overflow data to other data nodes. When a data node (called an **aggregators**) receives the data, it aggregates its own overflow data to reduce its size. After that, the aggregator forwards initiators' data to "visit" another data node, which then becomes an aggregator and aggregates its overflow data, and so on and so forth. This continues until enough aggregators are visited such that the total size of the data at data nodes after aggregation equals to or is slightly less than the total available storage in the network. A storage node can not be an aggregator since it does not have overflow data – when it receives the data, it simply relays it.

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 among all the data nodes, and to decide the sequence of aggregators/storage nodes to visit by each initiator, such that enough number of aggregators are visited while incurring minimum total energy consumption.*

Network Model. The sensor network is represented as an undirected connected graph $G(V, E)$, where $V = \{1, 2, ..., |V|\}$ is the set of $|V|$ uniformly deployed sensor nodes, and E is the set of $|E|$ edges. Two sensor nodes are connected by an edge if they are within transmission range of each other and thus can communicate directly (we assume that all the nodes have the same transmission range). 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 (our work can be extended to the cases wherein data nodes have different sizes of overflow data and/or storage nodes have different sizes of storage spaces). Due to the overall storage overflow, we have $p \times R > (|V| - p) \times m$, giving that $p > \frac{|V|m}{m+R}$ and $p \in \mathbb{Z}^+$.

Data Correlation Model [7]. We adopt an entropy-based spatial correlation model proposed in [7]. Let $H(X|Y)$ denote the conditional entropy of a random variable X given that random variable Y is known. Overflow data at data node i is represented as an entropy $H(i) = R$ bits if no side information is available from other data nodes; and $H(i|j_1, ..., j_p) = r \leq R$ bits, $j_k \in V_d \wedge j_k \neq i, 1 \leq k \leq p$, if data node i has available side information coming from at least another data node. That is, if a data node receives data from at least another data node, its overflow data can be aggregated, reducing the size from R to r. We are aware several entropy-based correlation models such as the ones proposed in [8, 33]. We adopt the one in [7] for the following two reasons. First, it is a simple and distributed coding strategy, which is easy to implement in sensor network application. Second, it is a realistic model since it approximates the case where the correlation function between two nodes decreases with their distance [7]. Consequently we make a few assumptions about the data aggregation in DRA:

Observation 1: Each data node can be either an initiator, or an aggregator, or none of them, but not both of them. An initiator can not be an aggregator because its data serves as side information for other nodes to aggregate. An aggregator can not 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. □

Observation 3: An initiator A can not be visited by another initiator B. This is because if so, 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] as the energy model for battery power consumption in wireless communication. In this model, for node u sending R-bit data to its one-hop neighbor v over their distance $l_{u,v}$, the transmission energy cost at u is $E_t(R, l) = E_{elec} \times R + \epsilon_{amp} \times R \times l_{u,v}^2$, the receiving energy cost at v is $E_r(R) = E_{elec} \times R$, which is independent of $l_{u,v}$. Here, $E_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $\epsilon_{amp} = 100pJ/bit/m^2$ calculates the energy consumption per bit on the transmit amplifier. Let $w(R, u, v) = E_t(R, l_{u,v}) + E_r(R)$, $w(R, u, v) = w(R, v, u)$. Let $W = \{v_1, v_2, ..., v_n\}$ be a sequence of n nodes with $(v_i, v_{i+1}) \in E$, $1 \leq i \leq n-1$ and $v_1 \neq v_n$. If all the nodes in W are distinct, W is a *path*; otherwise, it is a *walk*. Let $c(R, W) = \sum_{i=1}^{n-1} w(R, v_i, v_{i+1})$ denote the *aggregation cost* along W, which is the 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. [4]), so that the energy consumption contains only two parts: transmitting data and receiving data. Note that in this paper we do not consider how to upload data from storage nodes to base station (when uploading opportunities are available). Data mules or mobile data collectors can be used to upload data using techniques in [28] and [23].

Valid Range of Number of Data Nodes p for Feasible Overall Storage Overflow. We have shown the number of data nodes $p > \frac{|V|m}{m+R}$ due to overall storage overflow. To guarantee that the data after aggregation can fit in the available storage (i.e., a *feasible overall storage overflow*), we compute the upper bound of p next. Since there are p data nodes (each with R bits overflow data) and there are $|V| - p$ available storage nodes (each with m bits storage capacity), the data size

that needs to be reduced is $p \times R - (|V| - p) \times m = p \times (R + m) - |V| \times m$. Since each aggregator reduces its overflow data size by $(R - r)$, all together $\lceil \frac{p \times (R+m) - |V| \times m}{R-r} \rceil$ aggregators are needed. 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. Therefore we 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:

$$\frac{|V|m}{m+R} < p \leq \lfloor \frac{|V|m - R + r}{m + r} \rfloor. \quad (1)$$

Problem Formulation of DRA. Let

$$q = \lceil \frac{p \times (R+m) - |V| \times m}{R - r} \rceil \quad (2)$$

denote be the number of aggregators to visit. Given a valid p for feasible overall storage overflow, at most $(p - q)$ data nodes can be selected as initiators. The DRA selects:

- the set of a ($1 \leq a \leq (p - q)$) initiators from all the p data nodes V_d, denoted as $\mathcal{I}$, and
- the 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 and $W_j - \{I_j\} - G_j$ 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.

TABLE I
NOTATION SUMMARY

Notation	Explanation
V, $\|V\|$	The set and the number of sensor nodes
V_d, p	The set and the number of data nodes
q	The number of aggregators needed
m	The storage capacity of a storage node
R	The overflow data size at each data node before aggregation
r, $r < R$	The overflow data size at each data node after aggregation
$\mathcal{I}$, a	The set and the number of initiators, $1 \leq a \leq (p - q)$
I_j	The j^{th} initiator, $1 \leq j \leq a$
W_j	The aggregation walk starting with I_j
$w(R, u, v)$	The energy cost of sending R bits from u to its neighbor v
$c(R, W_j)$	The aggregation cost of sending R bits along W_j

The objective of the DRA is therefore to select a set of a initiators $\mathcal{I} \subset V_d$ and a set of a corresponding aggregation walks, each starting from a distinct initiator in $\mathcal{I}$, and each initiator sends its overflow data to all the aggregators in its corresponding aggregation walk, such that all together q aggregators are visited and therefore can aggregate their own overflow data, while 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. Table I lists all the notations.

EXAMPLE 1: Fig. 2 gives an example of DRA in a grid sensor network of nine nodes. Nodes B, D, E, G, and I are data nodes, while A, C, F and H are storage nodes. $R = m = 1$, $r = 3/4$, and energy consumption along any edge is 1. Overall storage overflow exists, since there are 4 units of storage while there are 5 units of overflow data. Number of aggregators q is calculated to be 4, leaving one data node to be initiator. One optimal solution could be selecting B as initiator and setting its aggregation walk as: B, E, D, G, H, I, with total aggregation cost of 5. □

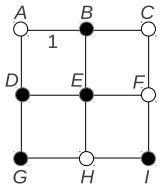


Fig. 2. An example of the DRA problem.

We find that solving DRA is equivalent to solving a new graph-theoretic problem, which we refer to as *multiple traveling salesman walks problem (MTSW)*. We first formulate and solve MTSW in Section III. In Section IV, we show that the DRA is equivalent to the MTSW in an appropriately transformed graph of the sensor network graph, therefore the algorithmic solutions of MTSW in Section III-B can be applied to solve the DRA.

III. Multiple Traveling Salesman Walks Problem (MTSW)

A. Problem Formulation and NP-Hardness.

Given an undirected weighted graph $G = (V, E)$ with $|V|$ nodes and $|E|$ edges,[2] 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 starting nodes (i.e., the initiator in DRA), from each of which a salesman can be dispatched to visit a number of other nodes following a walk, such that a) all together q nodes (excluding starting nodes) are visited, and b) the total cost of the walks is minimized. We have $b = |V| - q$.

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) is a special case of MTSW. TSW is defined as follows: Given a weighted graph with nonnegative edge weights, a starting node s and an ending node t, the goal is to find a minimum-length s-t walk that visits all vertices *at least* once. TSW is NP-hard (Section 6, [20]). The differences between TSW and MTSW are a) there could be multiple starting nodes in MTSW while is only one starting node in TSW, and b) the starting and ending nodes are not fixed in MTSW while s and t are given as input in MTSW. Therefore, when $b = 1$ in MTSW (i.e., only one of the $|V|$ nodes is allowed to dispatch a salesman), MTSW can be solved by calling TSW as a sub-routine upon

[2]Note that G is not necessarily a complete graph. Otherwise, a salesman does not need to visit a city more than once.

all possible pairs of s and t, and finding the one that yields the minimum cost. ∎

B. Algorithmic Solutions for MTSW

1) Linear Topologies: $G(V, E)$ 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 finds optimal solution for MTSW in linear topologies. It first finds the q edges in E with smallest weights, then checks if any of pair of them share an end node; if so, they belong to the same path.[3] Assume that it finally gets a disjoint paths. Then it starts at the leftmost node of each path to visit other nodes in this path. Therefore, in linear topology, each of the q nodes is visited exactly once.

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**:
   $W_1 = W_2 = ... = W_a = \phi$ (empty set);
   i: index for edges; j: index for paths;
1. $i = 1$; $j = 1$;
2. Find the first 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. $I_j = L(e_i)$; $W_j = \{L(e_i), R(e_i)\}$;
5. **while** $(i < q \wedge R(e_i) == L(e_{i+1}))$
6. $W_j = W_j \cup \{R(e_{i+1})\}$;
7. $i++$;
8. **end while;**
9. $i++$; $j++$;
10. **end while;**
11. $a = j$;
12. **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 $O(|E|\log q)$.

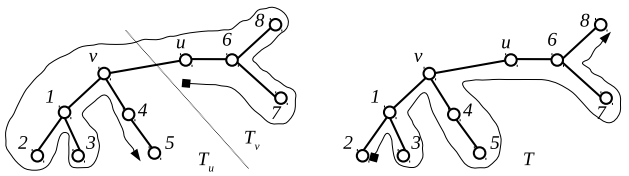
Theorem 2: Algorithm 1 is optimal for MTSW in linear topologies.

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 are 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. ∎

2) 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 the edge set $E' \subseteq E$, and for all $(u, v) \in E$, $u, v \in V'$ iff $(u, v) \in E'$. □

[3]In linear topologies, each obtained walk is a path, a sequence of distinct nodes.



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

Fig. 3. Illustrating walk in a tree.

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 $G[E']$). □

Binary Walk In A Tree. Assume that edge (u, v) has the maximum weight in a tree T (we randomly choose one if there are multiple edges with maximum weights). T then can be divided into (u, v) and two subtrees, rooted at u and v respectively. Denote them as T_u and T_v, as shown in Fig. 3(a). Let $c(T)$, $c(T_u)$, and $c(T_v)$ denote the cost of T, T_u, and T_v, respectively (i.e., the sum of weights of all edges). Below we define a cost-effective walk, called *binary walk (B-walk)*, which systematically visits all the nodes in T and serves as the building block for the approximation algorithm.

Definition 4: (**Binary Walk (B-Walk) In A Tree.**) Given edge (u, v) with maximum weight in tree T, the B-walk of T starts from u and visits all the nodes in T_u in a sequence following DFS and comes back, then visits v, from where it visits all the nodes in T_v in a sequence following DFS. Let $W_B(T) = \{u_1 = u, u_2, ..., u_n\}$ denote a B-walk and let $c(W_B(T)) = \sum_{i=1}^{n-1} w(u_i, u_{i+1})$ be its cost. □

Assume that in Fig. 3(a), $w(u, v) = 2$ while the weight of any other edges is 1. One of the B-Walks shown in Fig. 3(a) is: $u, 6, 7, 6, 8, 6, u, v, 1, 2, 1, 3, 1, v, 4, 5$, with a cost of 15. It can be seen easily that in a B-walk, (u, v) is traversed only once, each edge in T_u is traversed exactly twice, and each edge in T_v is traversed at most twice.[4] Note that for edges that are traversed twice in B-walk, their weights are counted twice towards to the cost of the walk.

Lemma 1: Let $|T|$ denote the number of edges in a tree T. We have $c(W_B(T)) \le (2 - \frac{1}{|T|}) \times c(T)$.

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

[4]In the special case that either u or v is a leaf node in T, (u, v) is traversed once and all other edges are traversed at most twice.

Approximation Algorithm For General Graph. Next we present an polynomial approximation algorithm (Algorithm 2), which yields a total cost of the walks that is at most $(2-\frac{1}{q})$ times of the optimal cost. It works as follows. Line 1 sorts all the edges in E into nondecreasing order of their weights. Line 2 initializes the set E_q to the empty set and creates $|V|$ trees, each containing one node. The while loop in lines 3-9 checks each edge (in the nondecreasing order of the weight), if it 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 all the connected components induced by these q edges. Since there is no cycles introduced during this process, each of those induced connected components could only be a linear or a tree topology. If it is linear, it starts from one end and visits the rest nodes in this linear topology exactly once; if it is a tree, it does a B-walk along the tree.

Algorithm 2: Approximation Algorithm For General Graphs.
Input: A general graph $G(V,E)$ and q, number of nodes to visit;
Output: set of a walks: $W_1, W_2, ..., W_a$, and $\sum_{1\le j\in a} c(W_j)$;
1. Let $w(e_1) \le w(e_2) \le ... \le w(e_{|E|})$;
2. $E_q = \phi$ (empty set), $i = j = k = 1$;
3. **while** $(k \le q)$
4. **if** (e_i is a cycleless edge w.r.t. E')
5 $E_q = E_q \cup \{e_i\}$;
6. $k++$;
7. **end if;**
8. $i++$;
9. **end while;**
10. Let $|C(G[E_q])| = a$;
11. **for** $(1 \le j \le a)$
12. **if** (C_j is linear) Starts from one end of C_j and visits the rest nodes in C_j;
13. **if** (C_j is a tree) Do a B-walk along C_j;
14. Let the resulted walk (or path) be W_j;
15. **end for;**
16. **RETURN** $W_1, W_2, ..., W_a$, and $\sum_{1\le j\in a} c(W_j)$.

Time Complexity and Discussions. Using disjoint-set data structure, the running time of Algorithm 2 is $O(|E|\log|E|)$. Algorithm 2 works similarly as finding edges of minimum spanning trees in the well-known Kruskal's algorithm [6], however, with two significant differences. First, instead of finding $|V|-1$ edges that connect all the nodes in V, Algorithm 2 only finds a set of q edges E_q, where $q \le |V|-1$. Therefore, instead of a single spanning tree that covers all the nodes in V resulted from Kruskal's algorithm, Algorithm 2 produces a *forest*, a graph with each connected component a tree. Second, unlike Kruskal's algorithm, which is an optimal algorithm, traveling each of the trees following a B-walk gives rise of an approximation algorithm for an NP-hard problem (Theorem 3).

Definition 5: (**Forest, q-Edge Forest, Cost of a Forest**) A forest F of G is a subgraph of G with each connected component a tree. A q-edge forest, denoted as F_q, is a forest with q edges. The cost of a forest F, denoted as $c(F)$, is the sum of weights of all edges in F, $c(F) = \sum_{e\in F} w_e$. □

The cost of the forest induced by E_q in Algorithm 2 is $c(E_q) = \sum_{e\in E_q} w(e)$ (here, we use E_q to also denote its induced forest when the context is clear). Next we prove that $c(E_q)$ is the minimum among the costs of all q-edge forests.

Lemma 2: $c(E_q) \le c(F), \forall F \in F_q$.

Proof: Let $E = \{e_1, e_2, ..., e_{|E|}\}$, with $w(e_1) \le w(e_2) \le ... \le w(e_{|E|})$. Let $E_q = \{e_1^g, e_2^g, ..., e_q^g\}$, with $w(e_1^g) \le w(e_2^g) \le ... \le w(e_q^g)$ (this is the order in which they are selected in Algorithm 2). By way of contradiction, assume that E_q is not a minimum-cost q-edge forest; instead $O_q = \{e_1^o, e_2^o, ..., e_q^o\}$ is a minimum-cost q-edge forest, with $w(e_1^o) \le w(e_2^o) \le ... \le w(e_q^o)$.

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. That is, $e_l^g \ne e_l^o$ and $e_i^g = e_i^o$, $\forall\ 1 \le i \le l-1$. According to Algorithm 2, $w(e_l^g) \le w(e_l^o)$. Now consider subgraph $O_q \cup \{e_l^g\}$. We have two cases.

Case 1: $O_q \cup \{e_l^g\}$ is a forest. Then $c(O_q \cup \{e_l^g\} - \{e_l^o\}) \le c(O_q)$, contradicting that O_q is a minimum cost 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 O_q is a forest. Next we claim that 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 $e_l^g = e_n, 1 \le n \le |E|$. We have two subcases.

Case 2.1: $e' \in \{e_1, e_2, ..., e_{n-1}\}$. To be exact, $e' \in \{e_1, e_2, ..., e_{n-1}\} - \{e_1^g, e_2^g, ..., e_{l-1}^g\}$. Thus $w(e') \le w(e_{n-1}) \le w(e_n) = w(e_l^g)$, contradicting that e_l^g and e_l^o are the minimum edges that differ.

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

Reaching contradiction in all the cases, it concludes that $c(E_q) \le c(F), \forall F \in F_q$. ∎

Let O be an optimal algorithm of MTSW, which gives the minimum cost of $\mathcal{O}$. Next we show that $c(E_q)$ is a lower bound of $\mathcal{O}$.

Lemma 3: $c(E_q) \le \mathcal{O}$.

Proof: Without loss of generality, assume that the edges selected in O induces o connected components, denoted as O_j $(1 \le j \le o)$. Assume that there are l_j nodes in O_j, and $s_j \ge 1$ of them are starting nodes (therefore there are s_j walks in O_j, visiting altogether $l_j - s_j$ nodes). Let the cost of the s_j walks in O_j be $c(W_j^o)$, $\sum_{j=1}^{o} c(W_j^o) = \mathcal{O}$.

Let $c(O_j)$ denote the sum of weights of all edges in O_j, $c(O_j) = \sum_{e\in O_j} w(e)$. We have $c(O_j) \le c(W_j^o)$ since each edge in O_j is traversed at least once in O. Next denote any tree of O_j as T_j^o, and denote the sum of all edges in T_j^o as $c(T_j^o)$. We have $c(T_j^o) \le c(O_j) \le c(W_j^o)$, resulting in $\sum_{j=1}^{o} c(T_j^o) \le \sum_{j=1}^{o} c(W_j^o) = \mathcal{O}$.

Let q' denote the total number of edges in the o connected components of O, $q' = \sum_{j=1}^{o}(l_j - 1)$. The subgraph induced by all T_j^o $(1 \le j \le o)$ is therefore a q'-edge forest. Since

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

Theorem 3: Algorithm 2 is a $(2 - \frac{1}{q})$-approximation algorithm for MTSW under general graph topologies, where q is number of distinct nodes to visit.

Proof: Algorithm 2 finds a connected components, C_j ($1 \leq j \leq a$), out of q selected edges E_q. 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 $c(E_q) = \sum_{j=1}^{a} c(C_j)$. For any C_j, let $c(W_j)$ denote sum of weights of all the edges traversed in B-DFS walk W_j. Following Lemma 1, $c(W_j) \leq (2 - \frac{1}{q_j}) \times c(C_j)$. Therefore, the total cost of the a walks found in Algorithm 2 is:

$$\begin{aligned}
\sum_{j=1}^{a} c(W_j) &\overset{\text{Lemma 1}}{\leq} \sum_{j=1}^{a} \left((2 - \frac{1}{q_j}) \times c(C_j) \right) \\
&< \sum_{j=1}^{a} \left((2 - \frac{1}{q}) \times c(C_j) \right) \\
&= (2 - \frac{1}{q}) \times c(E_q) \overset{\text{Lemma 3}}{\leq} (2 - \frac{1}{q}) \times \mathcal{O}.
\end{aligned}$$

∎

Smaller-Tree-First-Walk (STF-Walk). When a B-Walk traverses subtree T_u then T_v, each edge in T_u is traversed twice while each edge in T_v is traversed at most twice. Therefore, in order not to traverse many edges twice, a simple improvement could be to traverse, between T_u and T_v, the one with a smaller cost first. We refer to this as *smaller-tree-first-walk (STF-Walk)*.

Heuristic Algorithm For General Graphs. 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 as follows.

Definition 6: (**Longest-Path Walk (LP-Walk) In A Tree.**) Given a tree T, let $P = \{v_1, v_2, ..., v_n\}$ be the longest path in T. 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 exactly once. □

The novelty of the heuristic algorithm is based on the observation that when more edges are traversed only once, the cost of a walk can be further reduced. One of the LP-walks in the tree in Fig. 3(a) is now: $2, 1, 3, 1, v, 4, 5, 4, v, u, 6, 7, 6, 8$, as shown in Fig 3(b). It has a cost of 14. Because the maximum-weight edge (u, v) is not necessarily on the longest path P, we can not obtain performance guarantee for this heuristic algorithm. However, we show via extensive simulations in Section V that it outperforms the approximation algorithm by $20\% - 30\%$, in terms of energy consumption, under different network parameters.

IV. Algorithmic Solutions for DRA

Next we transform the original sensor network graph $G(V, E)$ into an aggregation graph $G'(V', E')$, which is defined below. We then show that solving DRA in G is equivalent to solving MTSW in G'.

Definition 7: (**Aggregation Graph.**) For a sensor network graph $G(V, E)$, its aggregation graph $G'(V', E')$ is defined as follows. V' is the set of p data nodes in V, i.e. $V' = V_d$. For any two data nodes $u, v \in V_d$ in G, there exists an edge $(u, v) \in E'$ in G' if and only if all the shortest paths between u and v in G do not contain other data nodes. 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. □

Theorem 4: DRA in $G(V, E)$ is equivalent to MTSW in $G'(V', E')$.

Proof: First, 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. As DRA concerns with visiting only 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 energy consumption of sending data from X to Y is accurately captured. This is so since the weight of the edge (X, Y) in G' represents the energy consumption along these shortest paths. In Fig. 2, (B, E), (D, E), (D, G), (E, I), and (G, I) belong to this case. Otherwise, if at least one of the shortest paths between data nodes X and Y contains other data nodes, edge (X, Y) is not included in G'. In Fig. 2, (B, I) belongs to this case among others.

Second, if there exists multiple shortest paths between two data nodes X and Y in G, some having at least another data node as intermediate nodes and some not, we argue that the intermediate nodes in the ones without data nodes are not included in the G'. In order to visit as many data nodes (aggregators) as possible while using as least amount of energy as possible, it mandates that DRA takes a shortest path with data nodes as intermediate nodes as part of the aggregation walk. (B, D) and (E, G) in Fig. 2 belong to this case.

Above rules guarantee that essential information in G for data aggregation, including data nodes and energy consumption sending data among them, are both accurately captured in the aggregation G'. Therefore, solving MTSW in G' is equivalent to solving DRA in G. ∎

Fig. 4(a) shows the aggregation graph G' of sensor network graph G in Fig. 2. Since 4 aggregators are needed ($q = 4$), using Algorithm 2, we find q-edge forest F of G' (Fig. 4(b)) and B-walk on F (Fig. 4(c)) sequentially.[5] Finally, we obtain the aggregation walk in G (Fig. 4(d)) by replacing each edge (u, v) in F with a shortest path between u and v in G (choose one randomly if there are multiple).

We note the fundamental difference between aggregation graph and 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 graph resembles the metric completion of graph in the definition of

[5]The LP-walk happens to be the same as B-walk in this example.

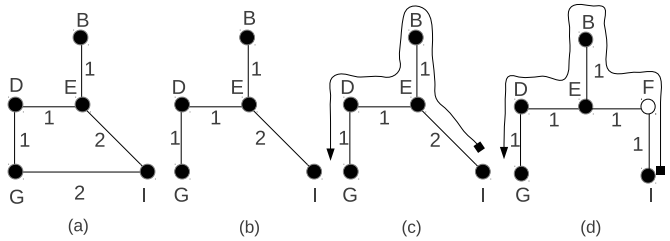
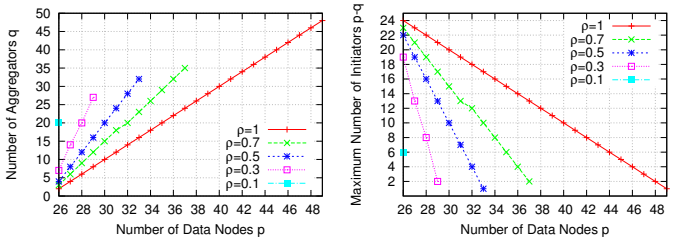


Fig. 4. (a) Aggregation graph G' of sensor network graph G in Fig. 2, (b) q-edge forest F from G', (c) B-walk (LP-walk) on F, and (d) Aggregation walk in G. The numbers on edges are their weights.



(a) Valid range of number of data nodes p, and number of aggregators q.

(b) Maximum number of initiators $p-q$.

Fig. 5. Feasible overall storage overflow with varying ρ ($R = m$).

the edge weight or length. However, aggregation graph of G is not necessarily a complete graph, due to the intrinsic multi-hop nature of a wireless sensor network.

V. Performance Evaluation

We compare the performance of the approximation algorithm (referred to as **B-Walk**) and the heuristic algorithm (referred to as **LP-Walk**). In our setup, 50 sensors are uniformly distributed in a region of $1000m \times 1000m$ square. Transmission range is $250m$; two sensor nodes can communicate directly if their distance is within the transmission range. Unless otherwise mentioned, the storage capacity of each storage node m is $512KB$, the size of overflow data at each data node R is $512KB$. We define *correlation coefficient* as $\rho = 1 - r/R$, where r is the size of overflow data after aggregation at each aggregator. The more correlation among data, the larger ρ is: $\rho = 0$ means no correlation at all, while $\rho = 1$ means perfect correlation (i.e., data at aggregators are duplicate copies of data at data nodes therefore can be completely removed). However, $\rho = 0$ is not considered in simulations since data correlation exists.

Feasible Overall Storage Overflow. Given $|V|, m, R, r$, Equation 1 gives the valid range of number of data nodes p for a feasible overall storage overflow. Given each valid p, Equation 2 finds its corresponding number of aggregators q that should be visited, therefore $p - q$ are the maximum number of allowable data nodes that can serve as initators. To investigate the feasible overall storage overflow, we study a sensor network with 50 nodes and set $R = m$.

Fig. 5(a) shows for different ρ, the valid range of p and the corresponding value of q for each value of p. When $\rho = 0.1$, the valid range of p is a single value of 26, with corresponding value of q 20. When increasing ρ, the valid range of p expands, from 26-29 in $\rho = 0.3$, to 26-33 in $\rho = 0.5$, to 26-37 in $\rho = 0.7$, to 26-49 in $\rho = 1$. This is because more data correlation leads to more data aggregation, thus more data nodes are allowed while satisfying feasible overflow condition. 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 need to be visited to achieve enough data size reduction.

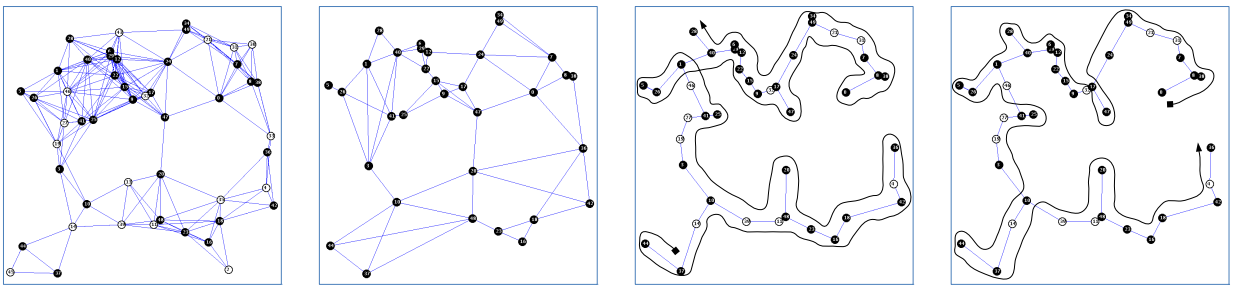
Consequently, less number of data nodes can serve as initiators, i.e., $p - q$ decreases with the increase of p, as shown in Fig. 5(b). It also shows that for the same p, $p - q$ increases with the increase of ρ. This is implied by Equation 2, which can be written as: $q = \lceil \frac{p \times (1 + m/R) - |V| \times m/R}{\rho} \rceil$. When p is fixed, more data correlation means that less number of aggregators are needed in order to reduce data size, therefore more data nodes can serve as initiators. Finally, Fig. 5(b) indicates that there are two cases in which only one initiator is allowed: $\rho = 0.5$ and $p = 33$, and $\rho = 1$ and $p = 49$, while multiple initiators are allowed for other cases.

Comparing SFT-Walk with B-Walk. We first compare SFT-Walk with B-Walk, where SFT-Walk traverses the smaller subtree first while B-Walk randomly chooses one of the two subtrees to traverse first. We choose $\rho = 0.5$ and vary p from 26 to 33. Fig. 6(a) shows that when p is small, both yield the same aggregation costs because the resulted trees are all linear topologies. However, when p gets larger, the aggregation cost of SFT-Walk is less than that of B-Walk. This is because SFT-Walk traverses the edges of the smaller subtree twice while B-Walk could possibly traverse the edges of the bigger subtree twice. Fig. 6(b) shows the performance improvement of SFT-Walk over B-Walk, which is the difference of their costs divided by the cost of B-Walk. In general, SFT-Walk improves B-Walk by $5 - 10\%$. We therefore adopt SFT-Walk for B-Walk for the rest of the simulations, and still refer it to as B-Walk.

Comparing B-Walk with LP-Walk Visually. Next we visually compare the performances of B-Walk and LP-Walk in a network of 50 nodes, for both cases of single initiator and multiple initiators.

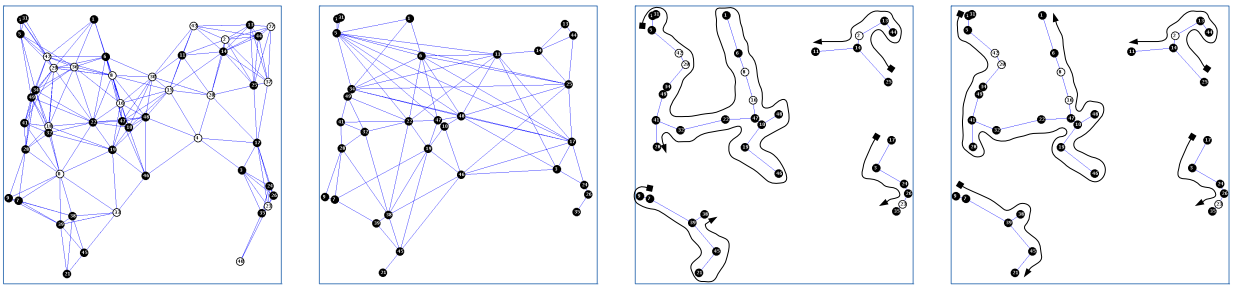
Single Initiator. When $\rho = 0.5$ and $p = 33$, $q = 32$ and $p - q = 1$. That is, there are 32 aggregators and one initiator. Fig. 7(a) and (b) show such a sensor network graph and its aggregation graph, respectively. Fig. 7(c) and (d) show the aggregation walks from B-Walk and LP-Walk, respectively. B-Walk visits 32 edges twice, resulting in a total 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. When $\rho = 0.5$ and $p = 32$, $q = 28$ and $p - q = 4$. That is, there are 29 aggregators and at most 4



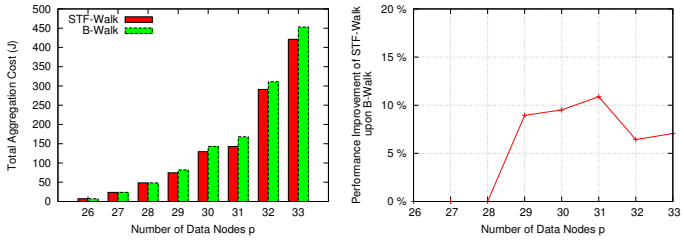
(a) Sensor network graph. (b) Aggregation graph. (c) B-Walk (cost=381.2J). (d) LP-Walk (cost=290.6J).

Fig. 7. Visually comparing B-Walk and LP-Walk with one initiator. ■ and ◄– indicate initiators and aggregators that are last visited.



(a) Sensor network graph. (b) Aggregation graph. (c) B-Walk (cost=255.9J). (d) LP-Walk (cost=203.0J).

Fig. 8. Visually comparing B-Walk and LP-Walk when 4 initiators are allowed.



(a) Total Aggregation Cost. (b) Performance Improvement of SFT-Walk Upon B-Walk.

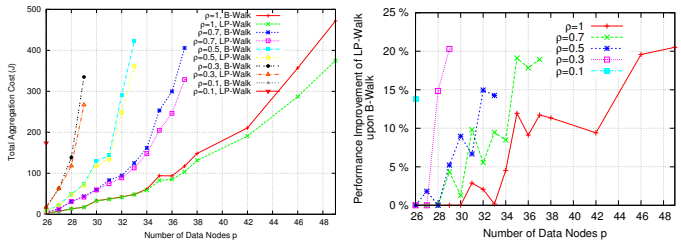
Fig. 6. Comparing STF-Walk and LP-Walk.

initiators. Fig. 8 shows that B-walk aggregation traverses 19 edges twice, resulting in a total cost of $255.9J$, while LP-Walk aggregation traverses 9 edges twice, with a total cost of $203.0J$, a 20.7% of improvement. Among the four trees in q-edge forest, B-Walk and LP-Walk find exactly the same aggregation walks in two smaller ones. This shows that when increasing number of initiators, the performance difference between B-Walk and LP-Walk gets smaller. This is because with more initiators, the resulted q-edge forest consists of more trees with smaller sizes, each with a "short" longest path. By traversing the edges on such short longest paths once, LP-Walk does not save as much at it can 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 can now be utilized to find more cost-effective aggregation walks.

Comparing B-Walk and LP-Walk by Varying p and ρ. Next we study the aggregation costs of B-Walk and LP-Walk, and the performance difference between them, by considering the whole ranges of p and ρ, that is, $\rho = 0.1, 0.3, 0.5, 0.7, 1.0$ and $p \in [26, 49]$. Fig. 9(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 perform better than B-Walk. It also shows that for the same p, with the increase of ρ, the aggregation costs for both B-Walk and LP-Walk decreases. This is because more correlation means that less number of aggregators need to be visited, reducing aggregation costs.

Fig. 9(b) calculates the performance improvement of LP-Walk upon B-Walk, which is defined as (total aggregation cost of B-Walk - total aggregation cost of LP-Walk)/ total aggregation cost of B-Walk. It shows that for each ρ, in each of its own valid range of p, the smaller the ρ, the larger of the performance improvement when p is fixed. For

example, when $p = 26$ (the only valid value for $\rho = 0.1$), the performance improvement 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 compared to $\rho = 0.5, 0.7, 1$. When less data correlation exists, more aggregators need to be visited, therefore the resulted q-edge forest gets larger as well as its constituent trees. By traversing the longest paths of larger trees once, LP-Walk can save more aggregation cost compared to traversing a smaller tree. This explains why LP-Walk has a larger performance improvement upon B-Walk when ρ gets smaller.

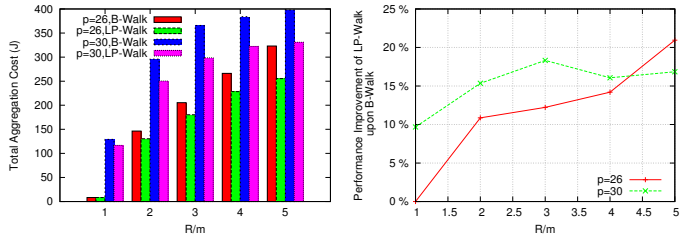


(a) Total Aggregation Cost.

(b) Performance Improvement of LP-Walk Upon B-Walk.

Fig. 9. Comparing B-Walk and LP-Walk by Varying p and ρ.

Comparing B-Walk and LP-Walk by Varying R/m. Finally, we compare the performances of B-Walk and LP-Walk based on different ratios of R/m. When increasing R/m, the overall storage overflow situations deteriorate since more overflow data needs to be accommodated. We set $\rho = 0.5$ and vary R/m from 1 to 5. The common range of p for different ratios of R/m is $[26, 30]$, therefore we pick 26 and 30 for p. Fig. 10(a) shows the total aggregation costs for both B-Walk and LP-Walk, when $p = 26$ and 30. Fig. 10(b) calculates the performance improvement of LP-Walk upon B-Walk, which is shown to increase generally when increasing R/m. This indicates that LP-Walk performs even better in more challenging overall storage overflow scenarios.



(a) Total Aggregation Cost.

(b) Performance Improvement of LP-Walk Upon B-Walk.

Fig. 10. Comparing B-Walk and LP-Walk by Varying R/m.

VI. Related Work

Below, we categorize and review the prior work in data resilience in sensor networks, data aggregation in sensor networks, and related graph-theoretic research.

A. Data Resilience in Sensor Networks.

Many data resilience techniques have been proposed to overcome against different causes of data loss in sensor networks. Ghose et al. [11] are among the first to propose Resilient Data-Centric Storage (R-DCS) to achieve resilience by replicating data at strategic locations in the sensor network. Ganesan [10] consider constructing partially disjoint multipaths to enable energy efficient recovery from failure of the shortest path between source and sink. Recently, network coding techniques are used to recover data from failure-prone sensor networks. Albano et al. [1] propose in-network erasure coding to improve data resilience to node failures. Kamra et al. [18] propose to replicating data compactly at neighboring nodes using growth codes that increase in efficiency as data accumulates at the sink. However, all these data resilience measures adopt the traditional sensor network model wherein base stations are always available near or inside the networks. Therefore, all the data resilience measures are designed towards transmitting data reliably to the base station.

Recently, some data resilience research has focused on how to preserve data in disconnection-tolerant storage sensor network in the absence of base station. We are aware of two lines of work in this direction. The first line is a sequence of system research [26, 27, 38, 44]. Since no base station is available, they design cooperative distributed storage systems specifically for disconnected operations of sensor networks, to improve the utilization of the networks data storage capacity. The other line of research instead takes an algorithmic approach by focusing on the hardness of the data preservation problems and the optimality of their solutions [14, 35, 43]. Tang et al. [35] address the energy-efficient data redistribution problem in data-intensive sensor networks. Hou et al. [14] study how to maximize the minimum remaining energy of the nodes that finally store the data, in order to store the data for long period of time. Xue et al. [43] consider that sensory data from different source nodes have different importance, and study how to preserve data with highest importance. However, all above work assume that there is enough storage space available from the network to hold all the overflow data and none of above work address the overall storage overflow problem.

B. Data Aggregation in Sensor Networks.

There is vast amount of literature of data aggregation in sensor networks. Here we only review the most recent and most related works. Tree-based routing structures are often proposed to either maximize the network lifetime (the time until the first node depletes its energy) [25, 41], or minimize the total energy consumption or communication cost [19, 22], or reduce the delay of data gathering [42]. In DRA, since the base station is not available and data must be stored inside

the network, tree-bases routing structure is no longer suitable. Instead, our data aggregation process follows a routing scheme that resembles traveling salesman problem [21]. Some works are based on non-tree routing structures and propose using mobile base stations to collect aggregated data in order to maximize the network lifetime [34, 36]. In contrast, we instead address a very different scenario for sensor networks: before the mobile base stations or data mules become available, some sensor nodes already deplete their storage, therefore their newly generated data must be offloaded to other sensor nodes with available storage. The challenge lies in the fact that the total generated data in the network overflows the total available storage in 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.

C. Overview of Related Theoretical Problems.

Below we give a brief overview of the well-known traveling salesman problem (TSP) [21] and the related vehicle routing problem (VRP) [37], which are both NP-hard, and identify the differences between these problems and the MTSW.

TSP asks the following question: Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city? A recent book by Gutin et al. [12] provide a compendium of results on the problem. The multiple traveling salesman problem (mTSP) [3] extends TSP to multiple salesman and determines a tour for each salesman such that the total tour cost is minimized and that each city is visited exactly once by one salesman. VRP refers to a whole class of problems involving the visiting of "customers" by "vehicles", wherein each customer has a positive demand and each vehicle has a limited capacity serving the demands. The goal of VRP is to find a route for each vehicle in order to supply all the customers and minimize the total cost of the routes. VRP is a generalization of the mTSP by considering demands from "customers" and "capacity" of the vehicles. Toth and Vigo [37] given an up-to-date survey of the variants and solution techniques for VRP.

The differences between mTSP/VRP and MTSW is as follows. Unlike mTSP, not only does MTSW need to figure out the order in which to visit cities, but it must answer some other fundamental questions: from which cities are salesmen dispatched and which cities does each salesman visit? In VRP, the set of vehicle locations (the depots) and the set of customer locations are usually disjoint. There is no such distinction for nodes in MTSW – each node can either dispatch a salesman (i.e. a vehicle) or be visited (i.e. as a customer). In VRP, a fixed set of customers (or cities) must be visited while in MSTW, it only requires all together q cities are visited, not necessarily a specific subset of cities. In VRP, each vehicle route starts and ends at the same or different depots while MSTW does not have such constraint.

The traveling salesman path problem (TSPP) [20] is defined as follows. Given an undirected graph $G = (V, E)$, a cost function on the edges, and two nodes $s, t \in V$, the TSPP is to find a Hamiltonian path from s to t visiting all cities exactly once (the case $s = t$ is equivalent to the TSP). Instead of a Hamiltonian path from s to t, the traveling salesman walk (TSW) problem [20] asks for the minimum cost $s-t$ traveling salesman walk, wherein the traveling salesman walk visits all vertices *at least* once. The MTSW we study is essentially a multiple traveling salesman walk problem, wherein up to some number of salesman can be dispatched from some cities, such that total q cities are visited with minimum amount of cost.

VII. Conclusion and Future Work

In this paper we solve overall storage overall problem in sensor networks by designing close-to-optimal data aggregation algorithms. Even though DRA is uniquely derived from sensor networks, it is a theoretically fundamental problem as well as a practical problem potentially with other applications. Its theoretical rigor lies in the underlying multiple traveling salesman walk problem, a new variation of the classic traveling salesman problem that has not been studied. Because of this theoretical root, the techniques proposed in this paper could be applicable not only in sensor networks, but in any applications in which data correlation and resource constraints coexist, such as scientific application, data centers, and big data analytics.

As future work, we will augment the proposed techniques to tackle situations with some nodes depleting their battery power, and design distributed algorithms for DRA. Currently the DRP is a static problem, in which the overflow data is generated at the beginning and only once. We will address a real-time problem where data is generated and transmitted dynamically and periodically, and investigate how the dynamics affect the problem and its solutions.

After being aggregated to the size accommodable by the available storage capacity, the overflow data need to be offloaded to storage nodes to be stored. Therefore the DRA is only the first stage of a more broad overall storage overflow in sensor networks. The second stage of offloading aggregated data from data nodes/aggregators to storage nodes has been studied extensively by existing research [14, 35, 43]. An interesting question to ask is: Should the data resilience measure be treated as two separate stages of data aggregation and then data offloading, or should it be treated in a holistic approach? In another word, is an optimal data aggregation plus an optimal data offloading optimal for the whole problem? The answer is no. 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 and future work, we would like to integrate these two stages together to explore a more energy-efficient solution for the overall storage overflow problem.

References

[1] Michele Albano and Jie Gao. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proc. of the International Workshop on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities (ALGOSENSOR'10)*, pages 105–117, 2010.

[2] Salah A. Aly, Zhenning Kong, and Emina Soljanin. Fountain codes based distributed storage algorithms for large-scale wireless sensor networks. In *Proc. of IPSN*, 2008.

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

[4] Costas Busch, Malik Magdon-Ismail, Fikret Sivrikaya, and Bulent Yener. Contention-free mac protocols for wireless sensor networks. In *Proc. of DISC*, pages 245–259, 2004.

[5] Harsha Chenji and Radu Stoleru. Mobile sensor network localization in harsh environments. In *Proceedings of the 6th IEEE international conference on Distributed Computing in Sensor Systems (DCOSS'10)*, pages 244–257, 2010.

[6] Thomas Corman, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

[7] 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.

[8] Rzvan Cristescu, Baltasar Beferull-lozano, Martin Vetterli, and Roger Wattenhofer. On network correlated data gathering. In *Proceedings of IEEE Infocom*, pages 2571–2582, 2004.

[9] John Heidemann Fred Stann. Rmst: Reliable data transport in sensor networks. In *Proc. of 1st IEEE International Workshop on Sensor Net Protocols and Applications (SNPA)*, pages 1–9, 2003.

[10] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, October 2001.

[11] Abhishek Ghose, Jens Grossklags, and John Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings the 4th International Conference on Mobile Data Management (MDM03*, pages 45–62, 2003.

[12] G. Gutin and A. Punnen, editors. *The Traveling Salesman Problem and its Variation*. Kluwer Academic Publishers, 2002.

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

[14] Xiang Hou, Zane Sumpter, Lucas Burson, Xinyu Xue, and Bin Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of IEEE 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] Jaein Jeong, Xiaofan Jiang, and D. Culler. Design and analysis of micro-solar power systems for wireless sensor networks. In *Proceedings of 5th International Conference on Networked Sensing Systems (INSS 2008)*, pages 181 – 188, 2008.

[17] Milica Stojanovic John Heidemann and Michele Zorzi. Underwater sensor networks: applications, advances and challenges. *Phil. Trans. R. Soc. A*, 370:158 – 175, 2012.

[18] Abhinav Kamra, Jon Feldman, Vishal Misra, and Dan Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proceedings of ACM Sigcomm*, 2006.

[19] Tung-Wei Kuo and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: Np-completeness and approximation algorithms. In *Proceedings of IEEE INFOCOM*, pages 2591 – 2595, 2012.

[20] Fumei Lam and Alantha 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] Jian Li, Amol Deshpande, and Samir Khuller. On computing compression trees for data collection in wireless sensor networks. In *Proceedings of INFOCOM*, pages 2115–2123, 2010.

[23] Ke Li, Chien-Chung Shen, and Guaning Chen. Energy-constrained bi-objective data muling in underwater wireless sensor networks. In *Proc. of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010)*, pages 332–341, 2010.

[24] Yunfeng Lin, Ben Liang, and Baochun Li. Data persistence in large-scale sensor networks with decentralized fountain codes. In *Proc. of INFOCOM*, 2007.

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

[26] 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.

[27] 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*.

[28] Ming Ma and Yuanyuan Yang. Data gathering in wireless sensor networks with mobile collectors. In *Proc. of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, pages 1–9, 2008.

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

[30] Ioannis Mathioudakis, Neil M. White, and Nick R. Harris. Wireless sensor networks: Applications utilizing satellite links. In *Proc. of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, pages 1–5, 2007.

[31] Univ. of Pittsburgh Pittsburgh PA USA ; Gadola G. Mosse, D. ; Comput. Sci. Dept. Controlling wind harvesting with wireless sensor networks. In *Proceedings of International Green Computing Conference (IGCC)*, pages 1 – 6, 2012.

[32] Luca Mottola. Programming storage-centric sensor networks with squirrel. In *Proceedings of the ACM/IEEE IPSN*, pages 1–12, 2010.

[33] Sundeep Pattem, Bhaskar Krishnamachari, and Ramesh Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(4):1–33, September 2008.

[34] Yi Shi and Y.T. Hou. Theoretical results on base station movement problem for sensor network. In *Proceedings of IEEE INFOCOM*, 2008.

[35] Bin Tang, Neeraj Jaggi, Haijie Wu, and Rohini Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2), May 2013.

[36] 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 *Proceedings of 18th International Workshop on Quality of Service (IWQoS)*, 2010.

[37] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 2001.

[38] Lili Wang, Yong Yang, Dong Kun Noh, Hieu Le, Tarek Abdelzaher, Michael Ward, and Jie Liu. Adaptsens: An adaptive data collection and storage service for solar-powered sensor networks. In *Proc. of the 30th IEEE Real-Time Systems Symposium (RTSS 2009)*.

[39] 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 *Proceedings of IEEE SECON*, pages 37 – 45, 2011.

[40] Geoff Werner-Allen, Konrad Lorincz, Jeff Johnson, Jonathan Lees, and Matt Welsh. Fidelity and yield in a volcano monitoring sensor network. In *Proc. of OSDI 2006*.

[41] 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 *Proceedings of IEEE INFOCOM*, pages 1566 – 1574, 2008.

[42] X. Xu, M. Li, X. Mao, S. Tang, and S. Wang. A delay-efficient algorithm for data aggregation in multihop wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22:163 – 175, 2011.

[43] Xinyu Xue, Xiang Hou, Bin Tang, and Rajiv Bagai. Data preservation in intermittently connected sensor networks with data priorities. In *Proc. of IEEE SECON 2013*, pages 65–73.

[44] Yong Yang, Lili Wang, Dong Kun Noh, Hieu Khac Le, and Tarek F. Abdelzaher. Solarstore: enhancing data reliability in solar-powered storage-centric sensor networks. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys), year = 2009,*.

[45] M. Z. Zamalloa and B. Krishnamachari. An analysis of unreliability and asymmetry in low-power wireless links. *ACM Transactions on Sensor Networks*, 3(2):1277–1280, 2007.