DRE^2: Achieving Data Resilience in Wireless Sensor Networks: A Quadratic Programming Approach

Shanglin Hsu, Yuning Yu, and Bin Tang
Department of Computer Science
California State University Dominguez Hills, Carson, CA 90747, USA
Email: {shsu10,yyu19}@toromail.csudh.edu, btang@csudh.edu

***Abstract*— We focus on sensor networks that are deployed in challenging environments, wherein sensors do not always have connected paths to a base station, and propose a new data resilience problem. We refer to it as DRE^2: *data resiliency in extreme environments*. As there are no connected paths between sensors and the base station, the goal of DRE^2 is to maximize data resilience by preserving the overflow data inside the network for maximum amount of time, considering that sensor nodes have limited storage capacity and unreplenishable battery power. We propose a quadratic programming-based algorithm to solve DRE^2 optimally. As quadratic programming is NP-hard and takes time to find the optimal solution, we design two time efficient heuristics based on different network metrics. We show via extensive experiments that all algorithms can achieve high data resiliences, while a minimum cost flow-based is most energy-efficient. Our algorithms tolerate node failures and network partitions caused by energy depletion of sensor nodes. Finally we study the feasibility of DRE^2, asking under which circumstances that data resilience is achievable. We put forward a maximum flow-based algorithm to solve it. Underlying our algorithms are flow networks that generalize the edge capacity constraint well-accepted in traditional network flow theory.**

Keywords – Data resilience, quadratic and linear programming, network flows, wireless sensor networks.

I. Introduction

Background and Motivation. *Data resilience* refers to the ability of any network to recover quickly and to continue maintaining availability of data despite of disruptions such as equipment failure, power outage, or malicious attack. Due to resource constraints challenges of wireless sensor networks such as unreplenishable battery power and limited storage capacity of sensor nodes [35], link unreliability and scarce bandwidth of wireless medium [41], and the inhospitable and harsh environments in which they are deployed [5, 6], sensor nodes are often prone to failure and vulnerable of data loss. Therefore, how to ensure that collected data reaches the base station reliably has been an active research topic since the inception of sensor network research. This line of research is usually named under the umbrella of data resilience [20], reliable data transmission [27], or data persistence [23]. We use data resilience throughout the paper.

However, all the existing data resilience research in traditional sensor networks assumes that a base station is always available to collect data, and focuses on how to encode and transmit data to the base station reliably. In this paper, we instead study data resilience from a totally different angle – from emerging sensor network applications wherein a base station is not available to collect the data. Such applications include volcano and seismic sensor networks [26], underground sensor networks [31], underwater or ocean sensor networks [9, 28], and volcano eruption monitoring and glacial melting monitoring [11, 32]. These emerging sensor network applications are designed and deployed to address some of the most fundamental problems facing human beings, such as disaster warning, climate change, and renewable energy.

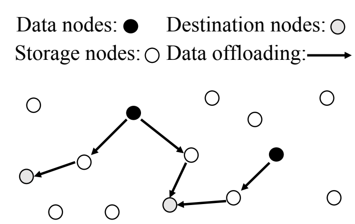


Fig. 1. The network model.

One common characteristic of these sensor networks is that they are all deployed in challenging or extreme 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. Consequently, it is not practical to deploy data-collecting base stations with power outlets in or near such inaccessible sensor fields. Sensory data generated thus has to be stored inside the network for some unpredictable period of time and then being collected by periodic visits of data mules or mobile sinks [7, 30]. Due to the lack of human intervention and the inadequacy of maintenance in the extreme environments, these sensing applications must operate more resiliently than traditional sensor networks.

Data Resilience Against Sensor Storage Overflow. In this paper, we focus on data resilience against sensor storage overflow, wherein storage spaces of some sensor nodes are depleted therefore they can not store any newly generated data [24, 35]. In our network model, shown in Fig. 1, there are some sensor nodes that are close to the event of interest and generate large amounts of sensory data. As those data cannot be relayed back to base stations in a timely manner due to the extreme environmental conditions, it must be stored locally and thus exhausts the limited storage capacity of these nodes. We refer to the sensor nodes that have exhausted their storage spaces as *data nodes*; the data that cannot be stored locally is referred to as *overflow data*. Other sensor nodes that have available storage are *storage nodes* (sensor nodes that generate data but still have available storages are considered as storage nodes).

In order to prevent data loss (we assume that any data loss means that the data resilience is not achieved), the overflow data at the data node must be offloaded to the storage nodes to be preserved, and then to be collected when above uploading opportunities become available. The storage nodes that finally store overflow data are *destination nodes*. We refer to this process wherein overflow data is offloaded from data nodes to destination nodes as *data offloading in sensor networks*. As it is not known beforehand when the next uploading opportunity arrives, it is preferred that the offloaded data being stored in destination nodes for longest amount of time before they run out of battery power. Assume that all the sensor nodes have the same energy depleting rates, data thus should be offloaded to destination nodes with high battery power. As each sensor node has limited storage capacity and battery power, the challenge is how to design data offloading scheme that maximizes the survival time for the data packets.

Contributions. Our contributions are twofold. On the practical side, we identify, formulate, and solve a new algorithmic problem in sensor networks called DRE^2: *data resiliency in extreme environments*. We accurately quantify data resiliences under limited storage capacity and unreplenishable battery power of sensor nodes, and design a quadratic programming (QP)-based algorithm to solve DRE^2 optimally (Section III). As QP is NP-hard and takes time for large scale networks, we design a suite of time-efficient and fault-tolerant heuristic algorithms (Section V). We show that all algorithms achieve high data resiliences while a minimum cost flow (MCF)-based algorithm is most energy efficient (Section VII). Finally we study a relevant and important problem called *feasibility of DRE*^{2} (Section VI). It asks under which conditions that all the overflow data packets can be offloaded into the network for preservation (i.e., achieving data resilience is feasible). We design a maximum flow (MF)-based algorithm to solve it. Both MCF and MF are formulated as integer linear programs (ILPs).

On the theory side, the underlying enabler of our techniques is flow networks that are delicately converted from the sensor network. These flow networks make possible to identify the convoluted relationship between energy consumption of sensor nodes and the flows of data offloading in DRE^2. As such, we find that in our flow networks, the relationship among flows, capacities, and costs on network edges are significantly different from those well-accepted in conventional network flows. In particular, we generalize the well-known *edge capacity constraint* of flow networks, which mandates that the number of flows on an edge is less than or equal to its capacity. In our designed flow networks, however, we propose that the capacity of an edge must be greater than or equal to the linear combination (i.e., weighted sum) of the flows on this edge. Such *generalized edge capacity constraint* uniquely arises in our flow networks and generalizes aforesaid widely used edge capacity constraint. With this generalization, we are able to apply QPs and ILPs on the flow networks to solve DRE^2 and its related feasibility problem optimally.

II. Related Work

Quadratic programming is the technique of optimizing a quadratic objective function with linear equality and inequality constraints [12, 13], and is one of the simplest forms of non-linear programming. It has been used in sensor network research to solve several important problems such as localization [10, 22], variational data assimilation problem for Lagrangian sensors [10], and optimized transmission for parameter estimation [36]. In contrast, we use this technique to solve a totally different problem. We believe our work is the first one to use quadratic programming to achieve optimal data resilience in sensor networks deployed in challenging environments.

Data resilience has been an active research since the inception of sensor network research. Ghose et al. [16] achieved resilience by replicating data at strategic locations in the sensor network. Ganesan et al. [14] constructed disjoint multipaths to enable energy efficient recovery from node failures. Recently, network coding techniques are used to recover data from failure-prone sensor networks. Albano et al. [4] proposed in-network erasure coding to improve data resilience to node failures. Kamra et al. [21] proposed to replicate data compactly at neighboring nodes using growth codes that increase in efficiency as data accumulates at the sink. As wireless sensor netowrks utilize sleeping mechanisms to conserve energy, which causes data availability problem, Xu et al. [38] proposed a dataset synchronization protocol in named data networking to achieve data resilience. However, all these research adopts the traditional sensor network model wherein base stations are always available near the networks, therefore are not suitable for the data resilience problem studied in this paper.

Some data resilience research has focused on how to preserve data in disconnection-tolerant sensor network in the absence of base stations. We are aware of two lines of work in this direction. The first line is a sequence of system research [25, 37, 40] that designed cooperative distributed storage systems to improve the utilization of the network's data storage capacity. The other line work is our own research. We took an algorithmic approach and designed a suite of data offloading techniques to achieve different objectives in sensor networks such as minimizing the total energy consumption [5, 35], maximizing the total priorities of preserved data [39], replicating data packets in the events of node failures [5, 34], as well as overcoming the overall storage overflow [33]. However, none of them addressed maximizing data resilience levels and the related feasibility problem. The closest work to ours is by Hou et al. [19], which achieves data resiliences by maximizing the minimum remaining energy of the destination nodes. Ours is to maximize the sum of the remaining energy of the destination nodes weighted by number of data packets stored on destination nodes, thus is different from their work.

III. Problem Formulation of DRE^2

Network Model. We model a sensor network as an undirected graph $G(V,E)$, where $V=\{1,2,...,n\}$ is the set of n nodes, and E is the set of m edges (two nodes are connected if their distance is within the sensor nodes' transmission range). There are l data nodes in the network, denoted as $V_d=\{1,2,...,l\}$. Data node $i \in V_d$ has d_i number of overflow data packets, each of k bits. The rest $n-l$ nodes are storage nodes, denoted as $V-V_d=\{l+1,l+2,...,n\}$. We denote the total $a=\sum_{i=1}^{l} d_i$ overflow data packets as $D=\{D_1,D_2,...,D_a\}$. Let the data node of D_j be $dn(j) \in V_d$. Let m_j be the available free storage space at storage node $j \in V-V_d$, meaning that j can further store m_j data packets. We assume naive feasibility condition that $a \le \sum_{j=l+1}^{n} m_j$ always holds. Otherwise, data loss is inevitable thus data resilience is not achieved.

Augmented Energy Model. We augment the well-known first order radio model [18] for wireless energy consumption. When node u sends a k-bit data packet to its one hop neighbor node v over their distance $l_{u,v}$ meters, the *transmission energy* spent by u is $E_u^t(v)=\epsilon_{elec}*k+\epsilon_{amp}*k*l_{u,v}^2$, the *receiving energy* spent by v is $E_v^r=\epsilon_{elec}*k$. Here $\epsilon_{elec}=100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $\epsilon_{amp}=100pJ/bit/m^2$ is the energy consumption per bit on the transmit amplifier.

However, this model does not take into account the energy consumption of storing data packets. As write operation costs 13.2μJ amount of energy in Toshiba 128MB flash [3] and we are dealing with large amounts of sensory data, we assume that energy consumption for storing data is non-negligible and augment above first order radio model with storing energy cost. In particular, when storing a data packet, a storage node $v \in V-V_d$ costs $E_v^s=\epsilon_{store}*k$ amount of *storing energy*. Here we assume that $\epsilon_{store}=100nJ/bit$ is the energy consumption when writing one bit on the memory of a sensor node. Let $E_{u,v}=E_u^t(v)+E_v^r$. We have $E_{v,u}=E_{u,v}$. Note a data node not only transmits all of its own data packets, but also can receive and transmit (i.e., relay) data packets for other data nodes. Meanwhile, a storage node can receive data packets from other nodes and then either transmits or stores them. Table I shows all the notations.

Problem Formulation. We define *offloading function* as $r: D \rightarrow V-V_d$, indicating that data packet $D_j \in D$ is distributed from its data node $dn(j) \in V_d$ to its destination node $r(j) \in V-V_d$. Let $P_j: dn(j),...,r(j)$, referred to as the *offloading path* of D_j, be the sequence of distinct sensor nodes along which D_j is offloaded from $dn(j)$ to $r(j)$. Let $\sigma(i,j)$ denote node i's successor node in P_j. Let $y_{i,j}$ be node i's energy cost of offloading data packet D_j, then

$$y_{i,j}=\begin{cases} E_i^t(\sigma(i,j)) & i=dn(j),\\ E_i^r+E_i^s & i=r(j),\\ E_{i,\sigma(i,j)} & i\in P_j-\{dn(j),r(j)\},\\ 0 & \text{otherwise.}\end{cases} \quad (1)$$

When i is the data node of D_j, it costs transmission energy $E_i^t(\sigma(i,j))$; when it is the destination node of D_j, it costs both receiving energy E_i^r and storing energy E_i^s; when it is a relaying node of D_j, it costs both receiving and transmission energy, the sum of which is $E_{i,\sigma(i,j)}$. Otherwise, node i is not involved in D_j's offloading thus costs zero amount of energy. Let E_i and $E_i^{'}$ denote sensor node i's initial energy level and remaining energy after all the a data packets are offloaded, respectively. Then, $E_i^{'}=E_i-\sum_{j=1}^{a} y_{i,j}, \forall\, i \in V$.

TABLE I
NOTATION SUMMARY

Notation	Description
V	The set of n data nodes
V_d	$V_d=\{1,...,l\}$ is the set of l data nodes, and $V-V_d=\{l+1,l+2,...,n\}$ is the set of storage nodes
d_i	Number of overflow data packets from data node $i \in V_d$
m_j	Storage capacity of storage node $j \in V-V_d$
D	$D=\{D_1,D_2,...,D_a\}$ is the set of a data packets
$dn(j)$	The data node of $D_j \in D$
E_i	Initial energy level of node i
$E_i^{'}$	Remaining energy level of node i after data offloading
$E_u^t(v)$	Transmission energy spent by u to transmit a packet to v
E_v^r	Receiving energy spent by v to receive one packet
E_v^s	Storing energy spent by v to store one packet
r	Data offloading function
P_j	The *offloading path* of data packet $D_j \in D$
$\sigma(i,j)$	Node i's successor node in P_j
$y_{i,j}$	Node i's energy cost of offloading data packet D_j
$\xi(i)$	Number of data packets stored at storage node i
$x_{i,j}$	The amount of flows on edge (i,j) in flow networks for QP and ILP
G'	$G'(V',E')$ is the flow network used for QP and MF ILP
G''	$G''(V'',E'')$ is the flow network for MCF ILP

Definition 1: (**Data Resilience Levels (DRLs).**) Given a sensor network G with a data packets to be offloaded, its *data resilience level* (DRL), denoted as $\mathcal{D}(G)$, is defined as the sum of remaining energy of the destination nodes of all the a data packet, i.e., $\mathcal{D}(G)=\sum_{j=1}^{a} E'_{r(j)}$. It is also the case that $\mathcal{D}(G)=\sum_{i=l+1}^{n}\left(E_i^{'}\times\xi(i)\right)$, where $\xi(i)$ is the number of data packets that are finally stored at storage node i. □

$\mathcal{D}(G)$ indicates the network's best achievable effort to preserve all a data packets, as the more energy of a storage node, the longer time its stored data can survive. The objective of DRE^2 is to find a offloading function r and a set of offloading paths $\mathcal{P}=\{P_1,P_2,...,P_a\}$ to offload the a data packets to their destination nodes, such that the DRL of the network is maximized after offloading, i.e., $\max_{r,\mathcal{P}} \mathcal{D}(G)$, under the energy constraint of sensor nodes: $E_i^{'} \ge 0, \forall\, i \in V$ and the storage capacity constraint of sensor nodes: $|\{j|r(j)=i, 1\le j\le a\}| \le m_i, \forall\, i \in V-V_d$.

EXAMPLE 1: Fig. 2(a) shows a linear sensor network with four nodes: 1 and 2 are data nodes, each has two overflow data packets; 3 and 4 are storage nodes, each has storage capacity of four. $E_4 \gg E_3$. To achieve maximum DRL, the optimal solution is to offload all the four data packets to node 4, even though it costs more energy than offloading to node 3. We use this example to illustrate our QP and ILP solutions next. □

IV. Quadratic Programming (QP) Solution

To maximize the DRL of any given instance of DRE^2, the fundamental challenge is to find each data packet's destination node as well as the offloading path from its data node to this

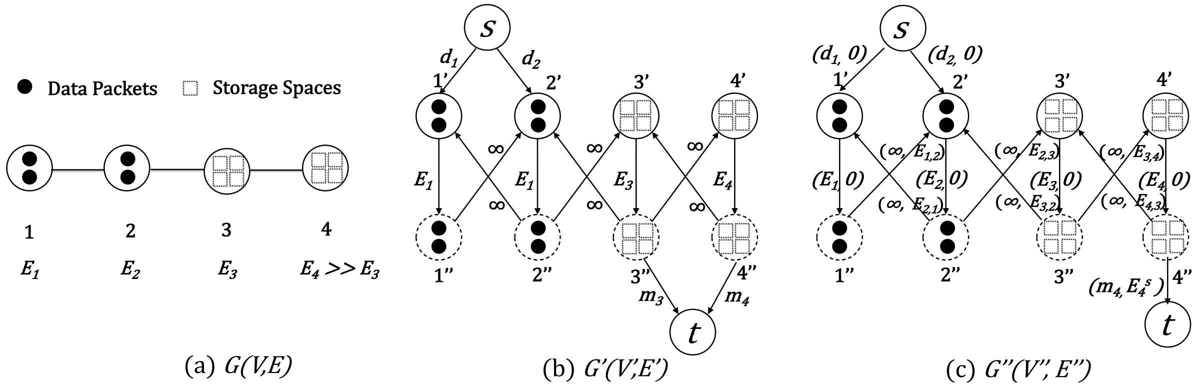


Fig. 2. (a) shows a linear sensor network $G(V, E)$ with two data nodes 1 and 2, each having two data packets to offload, and two storage nodes 3 and 4, each having four storage spaces. (b) shows its converted flow network $G'(V', E')$ for QA (A) that maximizes DRLs and ILP (C) that finds maximum number of data packets offloaded. (c) shows its converted flow network $G''(V'', E'')$ for ILP (B) that finds the minimum energy cost of data offloading. As $E_4 > E_3$ and node 4 has enough storage to store all the four data packets, the set of high-energy storage nodes $V_h = \{4\}$.

destination node. Then we are able to compute the number of data packets each destination node stores as well as its remaining energy level, therefore calculating the DRL. In particular, we need to represent energy levels, data packets, and storage capacities in a way that they can be computed – network flow modeling [29] is particularly suitable for such representation, as demonstrated below.

Graph Conversion. We first convert the sensor network graph $G(V, E)$ in Fig. 2(a) to a flow network $G'(V', E')$ in Fig. 2(b).

I. Replace each undirected edge $(i, j) \in E$ with two directed edges (i, j) and (j, i). Set the capacities of all the directed edges as infinity.
II. Split node $i \in V$ into two nodes: *in-node* i' and *out-node* i''. Add a directed edge (i', i'') with capacity of E_i, the initial energy level of node i. All incoming directed edges of node i are incident on i' and all outgoing directed edges of node i emanate from i''. Therefore the two directed edges (i, j) and (j, i) in Step I are now changed to (i'', j') and (j'', i').
III. Add a super source node s, and connect s to the in-node i' of the data node $i \in V_d$ with an edge. Set the capacity of this edge as d_i, the number of data packets at data node i.
IV. Add a super sink node t, and connect out-node i'' of the storage node $i \in V - V_d$ to t. Set its edge capacity m_i, the storage capacity of storage node i.

Hence, $V' = \{s\} \cup \{t\} \cup \{i' : i \in V\} \cup \{i'' : i \in V\}$ and $E' = \{(s, i') : i \in V_d\} \cup \{(i', i'') : i \in V\} \cup (i'', j') : (i, j) \in E\} \cup \{(j'', i') : (i, j) \in E\} \cup \{(i'', t) : i \in V - V_d\}$. We have $|V'| = 2n + 2$ and $|E'| = 2m + 2n$. Above conversion techniques are used in our previous work [19, 39] that solving related data preservation problems in sensor networks.

Rationale of the Conversion. The rationale of above conversion is fourfold. First, as the flows start from s and end at t in flow network G', and s connects to in-nodes of data nodes while t connects to out-nodes of storage nodes, it "forces" that overflow data packets are offloaded from data nodes to storage nodes. Second, with the node-splitting and the initial energy levels now being capacities of newly created edges, it guarantees that each node cannot spend more energy in data offloading than it has. Third, with the d_i and m_i now being "encoded" as the capacities of edges connecting s and i' (for data nodes) and i'' to t (for storage nodes), it makes sure that a data node cannot offload more than it has and a storage node cannot store more than its storage allows. Fourth, and most importantly, the energy consumption of each node can be accurately computed using flows in G', as shown below.

Computing Energy Consumptions of Nodes. Let $x_{i,j}$ be the amount of flows on directed edge (i, j) in G'. Recall that although both data nodes and storage nodes can receive data packets from other sensor nodes, the difference is that of all the received data packets, a data node must transmit all of them whereas a storage node can store some of them as long as its storage allows. Besides, a data node must transmit all of its own overflow data packets to others. Fig. 3(a) and (b) show the flows in G' that go through a data node and a storage node i respectively. We have below observations about the flows and their incurred energy cost, and the flow conservations.

- **Observation 1 (for both data and storage nodes).** Given any node $i \in V$ and any of its neighboring node j (i.e., $(i, j) \in E$)[1], the number of data packets i receives from j is $x_{j'',i'}$, the number of data packets i transmits to j is $x_{i'',j'}$. Thus, node i's total receiving energy cost is $E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'}$ and its total transmission energy cost is $\sum_{j:(i,j)\in E} (E_i^t(j) \times x_{i'',j'})$.
- **Observation 2 (for data nodes).** For any data node $i \in V_d$, the number of its own data packets that it transmits is $x_{s,i'}$. As the data packets i transmits are either its own or received from others, we have $x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'}$.
- **Observation 3 (for storage nodes).** For any storage node $i \in V - V_d$, the number of data packets it stores is $x_{i'',t} = \xi(i)$ (recall $\xi(i)$ is the number of data packets that are finally stored at storage node i). As the data packets i receives are either transmitted to other nodes or stored at i, we have $\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}$. The total storing energy cost of i is $E_i^s \times x_{i'',t}$.

Therefore the DRL can be represented as below:

[1]Note that it is not $(i, j) \in E'$, as all the data nodes, storage nodes, and neighboring nodes are sensor nodes in G, not in G'.

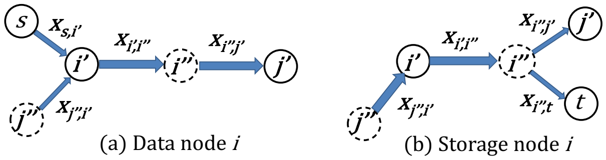


Fig. 3. (a) A data node transmits its own and relays data packets from others and (b) a storage node relays or stores the data packets from others.

$$\begin{aligned}\mathcal{D}(G) &= \sum_{i=l+1}^{n} \left(\xi(i) \times E_i^{'}\right) \\ &= \sum_{i \in V-V_d} \Bigg(x_{i'',t} \cdot \Big(E_i - E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} - \\ &\qquad \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) - E_i^s \times x_{i'',t} \Big) \Bigg). \end{aligned} \tag{2}$$

As $\mathcal{D}(G)$ is a concave quadratic expression, DRE^2 can thus be represented as below QP formulation (A):

$$(A) \qquad \max\ \mathcal{D}(G) \tag{3}$$

s.t.

$$x_{s,i'} = d_i, \quad \forall i \in V_d \tag{4}$$

$$x_{i'',t} \le m_i, \quad \forall i \in V - V_d \tag{5}$$

$$x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V_d \tag{6}$$

$$\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}, \qquad \forall i \in V - V_d \tag{7}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) \le E_i, \quad \forall i \in V_d \tag{8}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) + E_i^s \times x_{i'',t} \le E_i, \quad \forall i \in V - V_d \tag{9}$$

Eqn. 4 mandates that to achieve data resilience, data node i must be able to offload all its d_i number of data packets into the network (we will study the feasibility problem of DRE^2 in Sec. VI). Inequality 5 shows the storage constraint of storage nodes. Eqn. 6 and 7 show the flow conservation for data nodes and storage nodes, respectively (Observations 2 and 3). Inequalities 8 and 9 represent the energy constraint of data nodes and storage nodes respectively.

Generalized Edge Capacity Constraint. Inequalities 8 and 9 need some special notes. In Fig. 2(b) and 2(c), the amount of flows on edge $(i^{'}, i^{''})$ (i.e., $x_{i',i''}$) is not simply less than or equal to the capacity of $(i^{'}, i^{''})$ (i.e., E_i), as stipulated by the edge capacity constraint in conventional network flows. Instead, the relationship between $x_{i',i''}$ and E_i is more intricate, as shown in Inequalities 8 and 9. We observe that for data node i, $x_{i',i''}$ equals to the amount of data packets i can offload (i.e., $x_{s,i'}$) plus the amount of data packets it relays for other data nodes (l.h.s of Eqn. 6). $x_{i',i''}$ also equals to the total amount of data packets it transmits (r.h.s of Eqn. 6). As such, the energy cost of i can now be represented as a function of $x_{j'',i'}$ and $x_{i'',j'}$ (l.h.s. of Inequality 8). In other words, the total flows on (i', i'') are expressed as the linear combinations (i.e., weighted sum) of its constituent sub-flows. Similar observations can be made for storage node i, except the flow $x_{s,i'}$ in data node is now $x_{i'',t}$ in storage node. We refer to Inequalities 8 and 9 as *generalized edge capacity constraint*, and formally define it as below.

Definition 2: (**Generalized Edge Capacity Constraint.**) In a flow network, given any edge (u, v), let $f(u, v)$ and $cap(u, v)$ denote its flows and capacity respectively. The generalized capacity constraint stipulates that $\sum_{i=1}^{|f(u,v)|} a_i \le cap(u, v)$ where a_i is the weight for i^{th} flow on the edge. □

When $a_i = 1$ for all the flows, the generalized edge capacity constraint becomes $f(u, v) \le cap(u, v)$, the traditional edge capacity constraint. By generalizing this widely used constraint in flow network, we believe our work augments the network flow model and can have an impact on its related theory.

Solving QP. QP can be solved by the classic Wolfe's modified simplex method [13], which is based on solving a system of linear relations subject to complementarity conditions. There are many production QP solvers such as CGAL [1] and CPLEX [2]. We adopt CPLEX due to its performances. Besides, CPLEX can improve the efficiency of QP by allowing *gap tolerance* to find a feasible solution quickly (more in Section VII). As QP is NP-hard [15], we design two time-efficient heuristic algorithms below and show via experiments that they perform close to the optimal QP solution.

V. Heuristic Algorithms

To maximize DRLs, an intuitive solution is to offload data packets to nodes with initial high energy levels, defined below.

Definition 3: (**High-Energy Storage Nodes.**) High-energy storage nodes, denoted as V_h, are the set of storages nodes with the highest initial energy levels that can store all the a data packets. More formally, we sort storage nodes $V - V_d$ in non-ascending order of their initial energy: $E_{v_1} \ge E_{v_2} \ge ... \ge E_{v_{n-l}}$. Then the top $k+1$ nodes $\{v_1, ..., v_k, v_{k+1}\}$ where $\sum_{i=1}^{k} m_{v_i} < a \le \sum_{i=1}^{k+1} m_{v_i}$ is V_h. □

Both below algorithms are centered around how to offload data packets to V_h in an energy-efficient manner.

Network-Based Algorithm. For each storage node $i \in V_h$, Algo. 1 finds m_i data packets that are closest to i and offloads them to i via the currently available shortest path (in terms of energy consumption). Its time complexity is $O(n^2)$.

Algorithm 1: Network-Based Algorithm.

Input: A sensor network G with m_i, E_i, data packets D;

Output: $\mathcal{D}(G)$;

1. Compute $V_h = \{v_1, ..., v_k, v_{k+1}\}$;
2. **for** $(1 \le i \le k)$
3. Find the m_i data packets that are closest to v_i and offload them to v_i via the current shortest path between each data packet and v_i;

4. Update the energy levels of all the nodes on the path;
5. **end for;**
6. Offload each of the $a - \sum_{i=1}^{k} m_{v_i}$ data packets to v_{k+1} via shortest path and update the energy levels;
7. Compute $\mathcal{D}(G) = \sum_{i=l+1}^{n} \left(E_i^{'} \times \xi(i)\right)$;
8. **RETURN** $\mathcal{D}(G)$.

Minimum-Cost-Flow (MCF)-Based Algorithm. Although Algo. 1 saves energy by offloading data packets to their closest nodes in V_h, it does not consider global energy minimization in data offloading. Below we design Algo. 2 and prove that is minimizes total energy consumption in data offloading. It is a MCF-based algorithm applied on another properly converted flow network $G''(V'', E'')$ from the sensor network $G(V, E)$. In MCF, each edge in the flow network has a capacity and a cost and the goal is to minimize the total cost of the flows.

We first present the conversion and then the MCF ILP. As the first two steps of the conversion are the same as the one in Section IV, we start with Step IIII below:

III. For directed edge connecting s to the in-node i' of the data node $i \in V_s$, set its capacity as d_i and its cost as zero.
IV. For directed edge (i', i''), set its capacity as E_i and cost as zero.
V. For directed edge (i'', j'), set its capacity as infinity and cost as $E_{i,j} = E_i^t(j) + E_j^r$, the sum of node i's transmitting energy and node j's receiving energy. For (j'', i'), set its capacity as infinity and cost as $E_{j,i} = E_j^t(i) + E_i^r$, the sum of node j's transmitting energy and node i's receiving energy.
VI. For directed edge connecting the out-node i'' of the high-energy storage node $i \in V_h$ to t, set its capacity as m_i, the storage capacity of i, and its cost as E_i^s, the energy cost of storing one data packet by i.

With above transformation, the sensor network $G(V, E)$ in Fig. 2(a) is now converted to a flow network $G''(V'', E'')$ in Fig. 2(c). We next present the ILP formulation (B) for MCF.

$$(B) \qquad \min \sum_{(i,j)\in E''} x_{i,j} \times c_{i,j} \tag{10}$$

s.t.

$$x_{s,i'} = d_i, \quad \forall i \in V_d \tag{11}$$

$$x_{i'',t} \le m_i, \quad \forall i \in V_h \tag{12}$$

$$x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V_d \tag{13}$$

$$\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}, \quad \forall i \in V_h \tag{14}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} (E_i^t(j) \times x_{i'',j'}) \le E_i, \quad \forall i \in V_d \tag{15}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} E_i^t(j) \times x_{i'',j'} + E_i^s \times x_{i'',t} \le E_i, \quad \forall i \in V_h \tag{16}$$

$$\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V - V_d - V_h \tag{17}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} E_i^t(j) \times x_{i'',j'} \le E_i, \quad \forall i \in V - V_d - V_h \tag{18}$$

In the objective function 10, $x_{i,j}$ and $c_{i,j}$ are the amount of flows and cost on edge $(i, j) \in E''$, respectively. The Constraints 11-16 are similar to those in quadratic programming (A), except that Constraints 12, 14, and 16 are now applied on V_h, as only high-energy storage nodes V_h can store data packets. Finally, we add two more constraints viz. Equation 17 and Inequality 18 to respectively address the flow conservation and energy constraint of all the storage nodes that are not in V_h. Algo. 2 below calls ILP (B) as a subroutine:

Algorithm 2: MCF-Based Algorithm.
Input: A sensor network G with m_i, E_i, and D;
Output: $r : \mathcal{D}(G)$;
1. Compute $V_h = \{v_1, ..., v_k, v_{k+1}\}$;
2. Convert $G(V, E)$ to flow network $G''(V'', E'')$;
3. Compute ILP (B) on G'';
4. Compute $\mathcal{D}(G) = \sum_{i=l+1}^{n} \left(E_i^{'} \times \xi(i)\right)$;
5. **RETURN** $\mathcal{D}(G)$.

Theorem 1: Algo. 2 achieves minimum energy consumption in offloading a data packets to nodes in V_h.
Proof: The proof is omitted due to space constraint. ∎

Sovling MCF. We implement MCF ILP using CPLEX [2]. MCF can also be solved efficiently and optimally by combinatorial algorithms such as scaling push-relabel proposed by Goldberg [17]. Its time complexity is $O(a^2 \cdot b \cdot \log(a \cdot c))$, where a, b, and c are number of nodes, number of edges, and maximum edge capacity in the flow network, respectively.

All QP, Network- and MCF-based algorithms are fault-tolerant. Even with node failures and network partitions caused by energy depletion of sensor nodes, they can still achieve high DRLs and energy-efficiency as will be shown in Section VII.

VI. Feasibility Problem of DRE^2

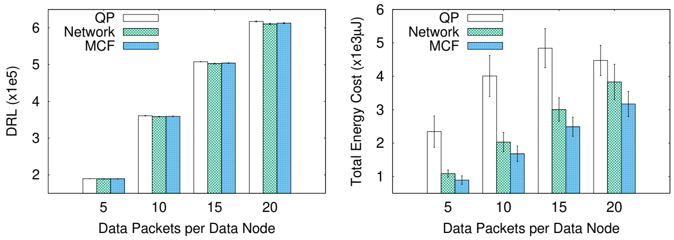
Due to aforesaid node failures and the resulted network partitions between data nodes and storage nodes, offloading all the data packets into the network is not always possible. For example, in Fig. 2(a), if E_3 is small enough, node 3 does not have enough energy to either store packets from nodes 1 and 2 or relay them to node 4, failing to achieve data resilience. We thus tackle an important *feasibility problem*: Given any instance of DRE^2, can all the a data packets be offloaded?

We answer this question by finding the maximum number of data packets offloaded, which is solved by below ILP (C) on the flow network $G'(V', E')$ in Fig. 2(b).

$$(C) \qquad \max \sum_{i\in V_d} x_{s,i'} \tag{19}$$

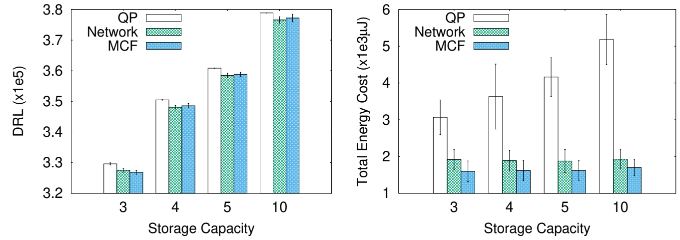
s.t.

$$x_{s,i'} \le d_i, \quad \forall i \in V_d \tag{20}$$



(a) Data resilience level. (b) Total Energy Cost.

Fig. 4. Small-scale comparison by varying d_i. $m_j = 5$.



(a) Data resilience level. (b) Total Energy Cost.

Fig. 5. Small-scale comparison by varying m_j. $d_i = 10$.

$$x_{i'',t} \leq m_i, \quad \forall i \in V - V_d \tag{21}$$

$$x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V_d \tag{22}$$

$$\sum_{j:(i,j)\in E} x_{j'',i'} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}, \quad \forall i \in V - V_d \tag{23}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right), \leq E_i \quad \forall i \in V_d \tag{24}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} E_i^t(j) \times x_{i'',j'} + E_i^s \times x_{i'',t} \leq E_i, \quad \forall i \in V - V_d \tag{25}$$

Objective Function 19 is to find the maximum flow; i.e., the maximum number of data packets that can be offloaded. All the constraints 20-25 are similar to those in QP (A), except that Eqn. 4 is changed to Inequality 20, as now it is not always possible to offload all the a data packets. We give below theorem regarding the feasibility of DRE^2.

Theorem 2: Given any instance of DRE^2 in G, if $\sum_{i\in V_d} x_{s,i'} = a$ in G', then achieving data resilience is feasible.

Proof: We have $\sum_{i\in V_d} d_i = a$. As $x_{s,i'} \leq d_i$ in G', we have $\sum_{i\in V_d} x_{s,i'} = \sum_{i\in V_d} d_i \leq a$. When $\sum_{i\in V_d} x_{s,i'} = a$, it must be that $x_{s,i'} = d_i$ for any $i \in V_d$, meaning each data node i successfully offloads its d_i data packets. Therefore all a data nodes are successfully offloaded. ■

Solving Maximum Flow. We implement maximum flow ILP using CPLEX [2]. Meanwhile, there are also two kinds of well-known combinatorial maximum flow algorithms viz. augmenting path and push-relabel [8]. Both algorithms are strongly polynomial while push-relabel is in general more flexible and efficient than augmenting path [29]. The time complexity of push–relabel maximum flow algorithm is $O(|V'|^2 \cdot |E'|)$ for a flow network $G(V', E')$.

VII. Performance Evaluation

We compare the performance of different algorithms viz. QP-based (referred to as **QP**), Network-Based (referred to as **Network**), and MCF-Based (referred to as **MCF**). We consider both small scale networks of 50 sensor nodes (10 of them are data nodes) and large scale of 100 nodes (20 of them are data nodes). The sensor nodes are uniformly distributed in a region of 1000m $\times$ 1000m. Each data node generates some number of data packets, each of 512B, that to be offloaded into the network. For initial energy levels, we consider both *varying model*, where different sensors have different initial energy, and *uniform model*, where all sensor nodes have the same initial energy level. Transmission range is 250m. In all plots, each data point is an average over ten runs, in each of which a different sensor network is generated; the error bars indicate 95% confidence intervals. For fair comparisons, in each run different algorithms use the same input including network topology, initial energy of each node E_i, the data nodes and their numbers of data packets d_i, and the storage nodes and their storage capacities m_j. For each sensor network, we first run feasibility checking – if not feasible, we generate another one and check again. All QP, MCF ILP and feasibility checking using maximum flow ILP are implemented in CPLEX [2].

Small-scale Comparison. As QP takes long time to run, we compare them in small scale of 50 nodes with 10 data nodes. The initial energy levels are random numbers in $[2000\mu J, 4000\mu J]$. Fig. 4(a) varies d_i from 5, 10, 15, to 20 while setting m_j as 5. It shows that with the increase of d_i, the DRLs achieved by all algorithms increase, as more data packets are now stored in storage nodes. QP always achieves slightly higher DRLs than MCF, while MCF higher than Network most of the time. As QP is optimal and all three perform close, this demonstrates the efficacy of all three algorithms in achieving data resilience. Fig. 4(b) shows the total energy consumptions of three algorithms, among which MCF has the smallest. QP costs the most energy, though, as it sometimes detours (instead of the shortest energy path) from sources to destinations in order to achiever high DRLs. Fig. 5 varies m_j while fixing $d_i = 10$. It shows again that QP achieves highest DRLs with the cost of energy consumption.

Tolerance Gap (%)	2	3	5	10	30
Execution time (sec)	2014.91	826.3	11.98	13.44	15.92
	788.8	182.53	8.27	7.89	7.98
	1034.27	160.98	22.64	22.8	28.32

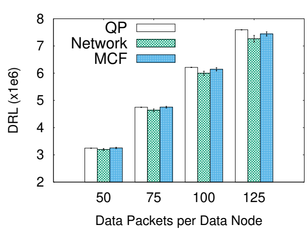
TABLE II
Investigating tolerance gap of the QP in CPLEX.

Large-scale comparison. Next we compare the algorithms in larger scale of 100 sensors, 20 of them are data nodes. As

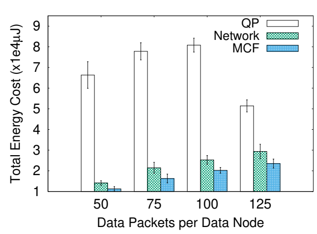
CPLEX [2] cannot finish computing an instance of QP after more than 13 hours, we decide to resort to below technique.

Gap Tolerance of the QP. To improve the time efficiency of QP, CPLEX [2] can be parameterized using a percentage value called *gap tolerance*. CPLEX stops once it finds a feasible solution within this percent of optimal. We thus investigate the tradeoff between time-efficiency and solution quality of different gap tolerances. Table II records the CPLEX execution time for different tolerance gaps between 2% and 30%, for three randomly generated networks. As 2% can take more than half an hour, we choose the next value of 3% as the gap tolerance for the QP; i.e., for the rest comparisons the QP always achieves at least 97% of the optimal DRLs.

Fig. 6 compare the three algorithms by varying d_i from 50, 75, 100, to 125 with $m_j = 50$ and $E_i = 2500\mu J$. We observe that even with fault tolerance, QP still outperforms the Network and MCF in DRLs. However, Fig. 6 (b) shows the energy cost of QP is much larger than those of the other two, as its focus is on high DRLs and not on the energy costs to achieve them. It also shows that the energy cost of QP decreases when d_i increases from 100 to 125. This is rather counter-intuitive, as offloading more data packets should cost more energy. Our conjecture for QP is that if there are multiple routes to offload a data packet without affecting the DRL maximization, it randomly chooses one as long as the gap tolerance level is met. When the network has more data packets to store, however, choosing such a random path could negatively affect the DRL maximization. As such, the QP begins choosing more energy-efficient offloading paths thus decreasing the energy cost. Fig. 7 varies m_j and shows that the performance differences of DRLs achieved by different algorithms seem to increase when increasing the storage capacity. As high energy nodes have more spaces to store data packets, the QP, being optimal, does a better job of utilizing the available spaces in order to maximize the DRL.

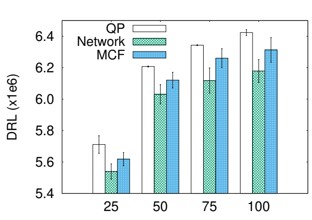


(a) Data resilience level.

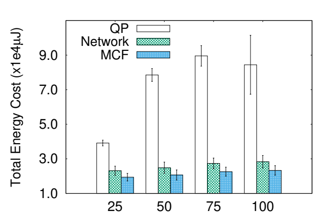


(b) Total Energy Cost.

Fig. 6. Large-scale comparison by varying d_i. $m_j = 50$.

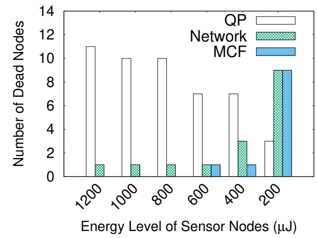


(a) Data resilience level.

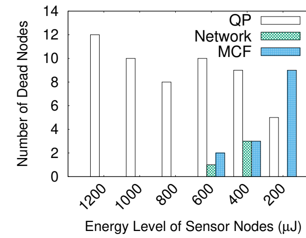


(b) Total Energy Cost.

Fig. 7. Large-scale comparison by varying m_j. $d_i = 100$.

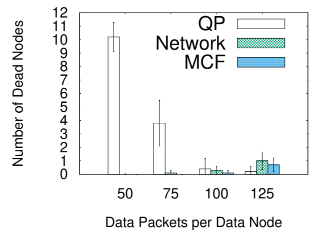


(a) $m_j = 50$.

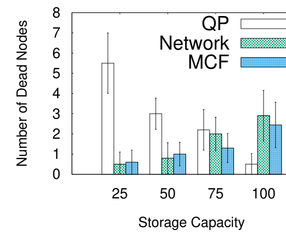


(b) $m_j = 13$.

Fig. 8. Fault-tolerance of three algorithms by varying E_i.



(a) Varying d_i. $m_j = 50$.



(b) Varying m_j. $d_i = 100$.

Fig. 9. Fault-tolerance of three algorithms at $E_i = 1200\mu J$.

Investigating Fault-Tolerance. Finally we investigate the fault-tolerance capability of the three algorithms by finding the number of dead nodes (i.e., nodes with depleted energy). It randomly generates one sensor network of 100 nodes, 20 of them are data nodes with $d_i = 50$. It starts by setting the initial energy levels E_i of all the nodes as $1200\mu J$, gradually decreases them, and records the number of dead nodes along the way for three algorithms. It stops until at least one of them fails to offload all the data packets. Fig. 8(a) sets m_j as 50 and shows all algorithms can tolerate up to around 10 node failures. However, as QP focuses more on DRLs and less on energy costs, it incurs more dead nodes than the other two most of the time. Fig 8(b) decreases m_j to 13, at which the network is almost full after data offloading, and shows that they can tolerate up to 6 and 8 node failures respectively before data loss occurs (note that when $E_i = 200\mu J$, Network cannot offload all the data packets). With the decrease of m_j, more storage nodes participate in the data offloading process (either storing or relaying), thus consuming more energy compared to when m_j is larger. Consequently, it allows a smaller number of dead nodes to take place before failing to offload all the data packets. Last, Fig. 9(a) and (b) study the fault-tolerance of algorithms at $E_i = 1200\mu J$ by varying d_i and m_j respectively. It is interesting to notice that with the increase of m_j, the number of dead nodes increases for both Network and MCF while decreasing for QP. For Network and MCF, as the number of destination nodes gets smaller with increase of m_j, less number of them participated in the data offloading process, depleting their energies more quickly. For QP, with the increase of m_j it can distribute data packets to nodes more evenly, thus reducing the number of dead nodes.

VIII. Conclusion and Future Work

We solved a new algorithmic problem called DRE^2 that achieved maximum data resilience inside sensor networks. It uniquely arises from emerging sensor network applications that are deployed in extreme enviromnets. We designed a QP-based optimal algorithm and two time- and energy-efficient heuristic algorithms viz. Network and MCF. We also solved the feasibility problem of DRE^2 by designing a maximum flow-based algorithm. Although sensor network research has been around for more than two decades, we believe that data resilience in our sensor network model for emerging applications has not been thoroughly addressed in existing literature. We uncovered a generalized edge capacity constraint model, wherein the consumed capacity on an edge is the linear combination of its flows. This generalizes the well-accepted edge capacity constraint in traditional network flows. As a future work, we will study if our heuristics can provide any performance guarantees in terms of DRLs. We plan to focus on a few specific topologies such as stars and trees, and investigate if the optimal and time-efficient algorithms exist.

Acknowledgment

This work was supported by NSF Grant CNS-1911191.

References

[1] Cgal 5.0.2 - linear and quadratic programming solver. https://doc.cgal.org/latest/QP_solver/index.html.

[2] Ibm cplex optimizer. https://www.ibm.com/analytics/cplex-optimizer.

[3] G. Aathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(4), 2009.

[4] M. Albano and J. Gao. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proc. of ALGOSENSOR*, 2010.

[5] B. Alhakami, B. Tang, J. Han, and M. Beheshti. Dao-r: Integrating data aggregation and offloading in sensor networks via data replication. *International Journal of Sensor Networks*, 29(2):134 – 146, 2019.

[6] Y. Chen and B. Tang. Data preservation in base station-less sensor networks: A game theoretic approach. In *Proc. of the 6th EAI International Conference on Game Theory for Networks (GameNets 2016)*.

[7] G. Citovsky, J. Gao, J. S. B. Mitchell, and J. Zeng. Exact and approximation algorithms for data mule scheduling in a sensor network. In *Proc. of ALGOSENSORS 2015*.

[8] T. Cormen, C. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second ed.)*. MIT Press and McGraw–Hill Press, 2009.

[9] R. W. L. Coutinho, A. Boukerche, and S. Guercin. Performance evaluation of candidate set selection procedures in underwater sensor networks. In *Proc. of IEEE ICC 2019*.

[10] L. Doherty, K. S. J. pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *Proc. of IEEE INFOCOM 2001*, 2001.

[11] S. Edwards, T. Murray, T. O'Farrell, I. C. Rutt, P. Loskot, I. Martin, N. Selmes, R. Aspey, T. James, S. L. Bevan, and T. Baugé. A High-Resolution Sensor Network for Monitoring Glacier Dynamics. *IEEE SENSORS JOURNAL*, 14(11):3926–3931, 2013.

[12] C. A. Floudas and V. Visweswaran. Quadratic optimization. *Nonconvex Optimization and Its Applications*, 2:217–269, 1995.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956. Web. 4 June 2015.

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

[15] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[16] A. Ghose, J. Grossklags, and J. Chuang. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proc. of MDM*, 2003.

[17] A. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.

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

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

[20] Y. Huang, J. F. Martínez, J. Sendra, and L. López. Resilient wireless sensor networks using topology control: A review. *Sensors (Basel)*, 15(10):24735–24770, 2015.

[21] A. Kamra, J. Feldman, V. Misra, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proc. of SIGCOMM*, 2006.

[22] J. Lee, W. Chung, and E. Kim. A new range-free localization method using quadratic programming. *Computer Communications*, 34(8):998–1010, 2011.

[23] F. Liu, M. Lin, Y. Hu, C. Luo, and F. Wu. Design and analysis of compressive data persistence in large-scale wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2685–2698, 2015.

[24] L. Liu and R. Wang. Message dissemination for throughput optimization in storage-limited opportunistic underwater sensor networks. In *Proc. of SECON 2016*.

[25] 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*.

[26] D. E. Phillips, M. Moazzami, G. Xing, and J. M. Lees. A sensor network for real-time volcano tomography: System design and deployment. In *Proc. of IEEE ICCCN 2017*.

[27] D. Puccinelli and M. Haenggi. Reliable data delivery in large-scale low-power sensor networks. *ACM Trans. Sen. Netw.*, 6(4):28:1–28:41, 2010.

[28] M. Rahmati and D. Pompili. Uwsvc: Scalable video coding transmission for in-network underwater imagery analysis. In *Proc. of IEEE MASS 2019*.

[29] K. A. Ravindra, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[30] R. Sugihara and R. K. Gupta. Path planning of data mules in sensor networks. *ACM Trans. Sen. Netw.*, 8(1):1:1–1:27, 2011.

[31] Z. Sun and I. F. Akyildiz. On capacity of magnetic induction-based wireless underground sensor networks. In *Proc. of INFOCOM*, 2012.

[32] Rui Tan, Guoliang Xin, Jinzhu Chen, Wen-Zhan Song, and Renjie Huang. Fusion-based volcanic earthquake detection and timing in wireless sensor networks. *ACM Transaction on Sensor Networks (ACM TOSN)*, 9, 2013.

[33] B. Tang. dao^2: Overcoming overall storage overflow in intermittently connected sensor networks. In *Proc. of IEEE INFOCOM 2018*.

[34] B. Tang, N. Jaggi, and M. Takahashi. Achieving data k-availability in intermittently connected sensor networks. In *Proc. of IEEE ICCCN 2014*.

[35] B. Tang, N. Jaggi, H. Wu, and R. Kurkal. Energy efficient data redistribution in sensor networks. *ACM Transactions on Sensor Networks*, 9(2):1–28, May 2013.

[36] A. Tinka, I. Strub, Q. Wu, and A. M. Bayen. Quadratic programming based data assimilation with passive drifting sensors for shallow water flows. In *Proc. of IEEE Conference on Decision and Control (CDC)*, 2009.

[37] L. Wang, Y. Yang, D. K. Noh, H. L., T. Abdelzaher, M. Ward, and J. 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)*.

[38] X. Xu, H. Zhang, T. Li, and L. Zhang. Achieving resilient data availability in wireless sensor networks. In *Proc. of IEEE ICC 2019 Workshops*.

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

[40] 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 *Proc. of MobiSys 2009*, pages 333–346, 2009.

[41] 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.