Priority-Based Data Preservation in Base Station-less Sensor Networks: a Maximum Weighted Flow Approach

Giovanni Rivera and Bin Tang
Department of Computer Science, California State University Dominguez Hills, Carson, CA, USA
Email: grivera64@toromail.csudh.edu, btang@csudh.edu

***Abstract*—Data sensed from the environment may have different importance and priority, and different data contributes distinctly to scientists' ability to analyze the physical environment. Meanwhile, many emerging sensing applications are deployed in challenging environments (e.g., underwater exploration and climate monitoring), wherein data-generating sensor nodes do not always have connected paths with the data-collecting base stations. When not all the data can be preserved inside such *base station-less sensor networks* (BSNs) due to severe energy and storage constraints at sensor nodes, how to preserve data to yield maximum total priorities energy-efficiently is a challenging new problem. We refer to this problem as the *priority-based data preservation* (PDP) and formulate it as a new graph-theoretical problem. Under a uniform-energy model, we uncover a new network flow problem called *maximum weighted flow* (MWF), which maximizes the total weight of flows in a flow network, considering different flows have different weights. MWF generalizes the classic maximum flow problem, wherein each flow unit has the same weight. Under a general energy model, we formulate the PDP as an integer linear program (ILP) and solve it optimally. We propose a greedy heuristic and a distributed PDP algorithm based on a push-relabel technique. The distributed algorithm yields maximum total priorities with provable time and message complexities. Extensive simulations show that our algorithms outperform the classic maximum flow algorithm (e.g., Edmonds-Karp) that does not consider data priority with up to 33.29% more total preserved priority. To our knowledge, this work is the first to study and solve the MWF.**

Keywords – Maximum Weighted Flow, Data Preservation, Data Priority, Base Station-less Sensor Networks, Energy-Efficiency

I. Introduction

Background and Motivation. Many of the emerging sensor network applications, such as underwater or ocean sensor networks [14, 23, 35, 42], volcano eruption monitoring, and glacial melting monitoring [31, 45], are deployed in remote or unattended areas. In those challenging environments, it is not feasible to have a long-term deployment of high-power data-collecting base stations (with power outlets) in the field. Consequently, sensory data generated in such environments is stored inside the network and uploaded when opportunities arise. These uploading opportunities could be periodic visits by data mules or robots [21, 46, 49], or autonomous underwater vehicles (AUVs) [4, 15], or transmission to the base station through wireless communication such as a low-rate satellite link [32]. Because the base station does not coexist with other sensor nodes in the field, we refer to such sensor networks as *base station-less sensor networks* (BSNs). The primary function of the BSN is to collect and store the generated sensory data inside the network between two uploading opportunities.

Meanwhile, with the strides made in sensor technologies over the past decade, sensor networks are evolving from dedicated application-specific platforms measuring single data types to integrated infrastructure measuring multiple data types simultaneously [24, 37]. Different data types could be collected within the same application, each contributing differently to the sensor network application. For example, in a sensor network that was deployed at Reventador (an active volcano in Ecuador) to monitor the volcano activities, the data collected includes seismic, infrasonic (low-frequency acoustic), and temperature [11, 45]. While all the collected data are helpful for scientists to understand and analyze the volcano activities, seismic and infrasonic data are more crucial than temperature data to interpret the key features of a volcano, such as its scale and magnitude. In modern sensor network applications, different data types have different importance. We refer to the importance of data as *data priorities*; the higher the priority, the more critical the data.

Challenges. In this paper, we focus on a challenging scenario in the BSN wherein its sensor nodes have reached a critical battery power and storage space level. This could happen after a BSN has been operated for a long time, and the uploading opportunities could not upload the data from the BSN promptly due to unpredictable factors such as inclement weather.[1] Meanwhile, when events of interest occur, sensors close to them may collect data more frequently than nodes far away, therefore running out of their storage space more quickly than others and cannot store newly generated data. Such data-generating nodes with storage depletion are called *data generators* (DGs). To avoid data loss, the *overflow data* (the newly generated data that can no longer be stored at DGs) must be offloaded to other sensor nodes with available storage spaces (referred to as *storage nodes*) to be preserved, so that

[1]We assume the battery power of sensor nodes is not replenishable, and we are aware of extensive research that studies how to recharge sensor nodes either wirelessly [17] or using solar power [43].

they can be uploaded when uploading opportunities become available. We call this process of offloading overflow data from DGs to storage nodes to be stored *data preservation in BSNs*. After this, robots and data mules can collect such preserved data when dispatched to the BSN (see a recent survey [22] for different techniques of how robots collect data from the sensor field). There could be data-generating nodes whose storage is not yet depleted - they are not considered DGs.

However, preserving all the overflow data in BSNs is not always possible due to the severe energy and storage constraints of sensor nodes. When there is insufficient storage, some overflow data must be discarded in the data preservation process, causing data loss. When there is not enough battery power for some sensor nodes, they could soon deplete their energy, causing network partition in the BSN and blocking the data preservation from the DGs to the storage nodes. Consequently, only part of the overflow data could be successfully preserved due to severe energy and storage constraints of sensor nodes. Considering that different data have different priorities, from the perspective of the BSN network operator and domain scientists, finding a data preservation process that maximizes its total preserved priority is critical. As data preservation is energy-expensive wireless communication, finding an energy-efficient preservation process that maximizes its total preserved priority becomes a new challenge.

Our Contributions. We tackle this challenge by establishing a new algorithmic framework called *priority-based data preservation* (PDP). Underlying the framework is a suite of new graph-theoretical problems called PDP problems. We study the PDP from the network flow perspective and design a suite of network flow-based algorithms. Network flow problems [3], such as *maximum flow, minimum cost flow, and multi-commodity flow*, study how to move objects (e.g., goods, vehicles, and network packets) around in a network efficiently following some resource constraints of the network. Coupling deep theoretical rigor and remarkable applicability, network flow has found great success in various applications in computer science, operations research, and engineering [28].

The PDP gives rise to a new network flow problem, which we refer to as *maximum weighted flow* (MWF). MWF aims to maximize the total weight of flows in a network, considering different flows could have different weights. MWF generalizes the classic maximum flow problem, wherein each flow unit has the same weight, and the goal is to maximize the total number of flows in the network. We find that under a uniform energy model wherein the sensor nodes cost the same energy when preserving a data item, MWF achieves maximum flow and yields the maximum total weight of all the flows; in contrast, a classic maximum flow is not necessarily an MWF. Under a more general energy model (i.e., wireless first-order radio model [18]) wherein the energy consumption of sensor nodes is different, we show that MWF does not need to be a maximum flow to maximize the total weight of all the flows. Our PDP model, with its theoretical roots in network flows and the new theory uncovered, may apply to other network applications wherein flow priorities are concerned, such as flow control in modern data centers where data flows have different priorities [33]. It is a very simple but general information producer and consumer model that has yet to be adequately explored in any other context. This paper's main contributions and organization are as follows.

1). We formally formulate the PDP in BSNs and review all the related work. (Section II)
2). We formulate the MWF and design an optimal algorithm to solve it efficiently. To our knowledge, the MWF has not been studied in any of the existing literature. (Section III)
2). We show that the PDP under a uniform energy model is equivalent to the MWF in a flow network properly transformed from the BSN graph. Thus, PDP can be solved optimally and efficiently. (Section IV)
3). Under a more general energy model, we formulate it as an Integer Linear Program (ILP) and solve PDP optimally. We also design an efficient greedy algorithm to solve the PDP. (Section V)
4). We design a distributed data preservation algorithm based on classic push-relabel maximum flow algorithm [16] and show it achieves optimal data priorities. Its time and message complexities are $O(kn^2)$ and $O(n^2m)$, respectively, where n, m, and k are the number of nodes, edges, and data generators. (Section VI)
5). Via extensive simulations, we show that our priority-based data preservation algorithms outperform the classic maximum flow algorithms that do not consider data priority (e.g., Edmonds-Karp [9]) by yielding up to 33.29% more total preserved priority. We also show that the distributed data preservation algorithm performs better than the classic push-relabel algorithm that does not consider the flow priorities under different network parameters. (Section VII)

II. Priority-Based Data Preservation Problem (PDP)

This section formally defines the PDP and then discusses related work to contextualize our paper's contributions.

A. Problem Formulation of PDP

Network Model. The BSN is represented as an undirected connected graph $G(V, E)$, where $V = \{1, 2, ..., n\}$ is n uniformly deployed sensor nodes, and E is the set of m edges. An edge connects two sensor nodes if they are within transmission range of each other and thus can communicate directly. There are k data generators (DGs), denoted as $V_d = \{DG_1, DG_2, ..., DG_k\} \subset V$. The rest of the nodes are storage nodes, denoted as V_s. According to our definition, DGs are storage-depleted; therefore, sensor nodes without depleted storage are not considered DGs. The sensory data are modeled as a sequence of raw data items, each of which is of unit size (our work can be easily extended to the case where data items have different sizes). Data items generated at DG_i all have priority value v_i, indicating their importance in a specific sensor network application. Let d_i denote the number of data items DG_i needs to be preserved (the amount of overflow data

at DG_i). Let m_i be the available free storage space (in terms of number of data items) at sensor node $i \in V$. If $i \in V_d$, then $m_i = 0$, implying that a DG node is storage-depleted and thus has zero available storage space. If $i \in V_s$, then $m_i > 0$, implying that storage node i can store another m_i data items. Let $q = \sum_{i=1}^{k} d_i$ be the total number of data items to be preserved in the BSN. Note it could be that $\sum_{i \in V_s} m_i \geq q$ or $\sum_{i \in V_s} m_i < q$ and our proposed techniques apply to both.

Energy Model. Sensor node i (including DGs and storage nodes) has finite and unreplenishable initial energy E_i. We consider two energy models.

- *Uniform Energy Model.* In this model, any node involved in preserving a data item, including sending, receiving, or relaying the data item, costs the same amount and one unit of energy. That is, the energy consumption of distributing a data item from its DG to a storage node equals the number of nodes involved in the data distribution. We believe this is a good approximation of the energy consumption for uniformly deployed sensor nodes.
- *General Energy Model.* The first-order radio model [18] is a more general energy model for wireless communication. For k-bit data over distance l, the transmission energy incurred on the sender is $E_{Tx}(k,l) = E_{elec} \times k + \epsilon_{amp} \times k \times l^2$, and the receiving energy incurred on the receiver is $E_{Rx}(k) = E_{elec} \times k$. 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.

Problem Formulation. Let $D = \{D_1, D_2, ..., D_q\}$ denote the set of q overflow data items in the entire network. Let $s(j) \in V_d$, where $1 \leq j \leq q$, denote D_j's DG. The PDP decides:

- the set of data items $\mathcal{D} \subseteq D$ to preserve, and
- *preservation function* $r : \mathcal{D} \rightarrow V_s$, indicating that data item $D_j \in \mathcal{D}$ is offloaded from $s(j)$ to its *destination node* $r(j) \in V_s$, and
- *preservation path* of $D_j \in \mathcal{D}$, referred to as $\mathcal{P}_j : s(j), ..., r(j)$, a simple path (i.e., a set of distinct sensor nodes) along which D_j is offloaded from $s(j)$ to $r(j)$.

Let x_{ij} be the energy cost incurred by sensor node i in distributing D_j from $s(j)$ to $r(j)$, and let E'_i denote i's energy level after all the $|\mathcal{D}|$ data items are preserved. Then, $E'_i = E_i - \sum_{D_j \in \mathcal{D}} x_{ij}, \forall\ i \in V$. We assume that a node can still relay data items even though it has full storage. The objective of PDP is to select data items $\mathcal{D} \subseteq D$ to preserve and find corresponding preservation path $\mathcal{P}_j$ of $D_j \in \mathcal{D}$ to offload them to their destination nodes, such that the total priorities of the preserved data is maximized, i.e., $\max_{\mathcal{D}} \sum_{D_j \in \mathcal{D}} v_{s(j)}$, under the energy constraint that each node can not spend more energy than its initial energy level, $E'_i \geq 0, \forall\ i \in V$, and the storage capacity constraint that the number of data items offloaded to destination node i is less than or equal to i's storage capacity, $|\{j \mid r(j) = i, D_j \in \mathcal{D}\}| \leq m_i, \forall\ i \in V_s$. Table I lists all the notations.

EXAMPLE 1: Fig. 1 gives a small linear BSN with four nodes. Nodes 1 and 2 are DGs, each with two overflow data items to preserve and an initial energy level of 1. Nodes 3 and 4 are storage nodes, with 2 and 2 available storage spaces, respectively, and each has an initial energy level of 2. The priority of each data in node 1 is 3, and 1 in node 2. The optimal PDP solution is that one data item of node one is offloaded to node 3, resulting in a preserved priority of 3. The other solution is that one data item of node 2 is offloaded to node 3, resulting in a preserved priority of 1. □

TABLE I: Notation Summary

Notation	Explanation
V	The set of sensor nodes in the BSN
E_i	The initial energy of node i
m_i	The storage capacity of node i
E'_i	The remaining energy of node i after data preservation
V_d	The set of data generators (DGs)
V_s	The set of storage nodes
D	The entire set of data items in the BSN
$\mathcal{D}$	The set of data items selected to be preserved
DG_i	The i^{th} DG, $1 \leq i \leq k$
D_j	The j^{th} data item
d_i	The number of overflow data items at DG_i
q	The total number of overflow data items, $q = \sum_{i=1}^{k} d_i$
v_i	The priority of each data item at DG_i
x_{ij}	The energy cost of node i in preserving D_j
$s(j)$	The DG node of data item D_j
$r(j)$	The destination node of D_j; it is a storage node
$\mathcal{P}_j$	The preservation path of D_j

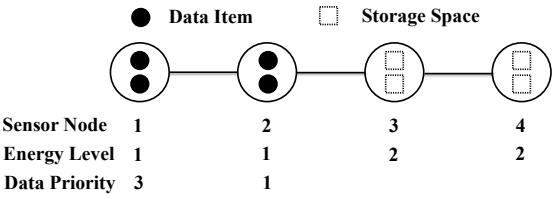


Fig. 1: Illustration of the PDP problem.

B. Related Work

We first review all the data preservation research in BSNs. As data preservation is about data gathering in sensor networks, we also review some of the traditional data-gathering techniques and paradigms in sensor networks to give a context to the contributions made by our research. We finally review the priority-based techniques in both theory and sensor network communities.

Data Preservation in the BSN. Data collection and storage in BSNs started as a series of system research [29, 30, 44]. Luo et al. [30] proposed a new cooperative storage system for sensor networks geared for disconnected operation (where sensor nodes do not have a connected path to a base station). The goal of this system, called EnviroStore, is to maximize its data storage capacity by opportunistically offloading data to external devices when possible. In their following works, they designed a distributed acoustic monitoring, storage, and trace retrieval system to study animal populations in the wild [29] and extended their system for reliable data collection

using solar-powered sensor networks [44]. As they are system research that focuses on system prototyping, all the data collection mechanisms proposed are heuristic-based and need more rigorous performance analysis and guarantees.

Inspired by the above system research, another line of research took a more formal algorithmic approach. It formulated data preservation problems in BSNs as a suite of graph-theoretical problems [20, 38–41, 47]. Tang et al. [40] showed that energy consumption minimization in data preservation is a minimum-cost flow problem. Hou et al. [19] and Hsu et al. [20] studied how to maximize the minimum remaining energy and the sum of the remaining energy of the nodes that finally store the data, such that the data can be preserved for the maximum amount of time. Tang et al. [39, 41] considered a particular case in data preservation called *overall storage overflow*, wherein there is insufficient available storage in the BSN to store all the overflow data. They proposed data aggregation to solve the overall storage overflow and identified a new graph-theoretical problem called the traveling salesman placement problem. Rivera et al. [38] studied the performance of Nash Equilibria for data preservation in BSN, and Yu et al. [50] motivated the selfish sensor nodes to achieve truthful and optimal data preservation using an integrated game theory and network flow approach.

However, all the above works assume that the sensory data in the BSN can all be successfully preserved (offloaded from DGs to storage nodes), and none consider data priorities. In more challenging scenarios, when nodes reach severely low energy levels, and not all the data packets (with different priorities) can be preserved, how to preserve the data with the total maximum priorities becomes a challenging new problem, which is the topic of this paper. The conference version of this paper [48] identified data preservation under the uniform energy model as the maximum weighted flow problem and solved it optimally. This paper extended it by considering a more general energy model.

Delay Tolerant Networks and Compressive Sensing. BSNs resemble delay tolerant networks (DTN) [7] and compressive sensing [8, 13, 36] in that they are all related to how sensor measurements are transmitted and stored. However, BSNs and DTNs fundamentally differ in mobility models and objectives. First, in DTNs, mobile nodes are intermittently connected due to their mobility and low density, and relay nodes opportunistically forward data to destination nodes; in a BSN, all the static sensors are connected while being disconnected from the base station, and data is uploaded to the base station only when uploading opportunities are available. Second, the research objectives in DTNs are mainly to increase the data delivery rate or reduce data delivery delay, whereas, in BSNs, the goal is to preserve the most valuable data for a maximum amount of time. Compressive sensing [8, 13, 36], on the other hand, offers a new perspective for distributed sensor networks for energy-efficient and low-cost data acquisition and transmission. Instead of transmitting the original sensory data, it computes a weighted average of sensor measurements in a low-dimensional and incoherent space. However, all the compressive sensing research in sensor networks still assumes the base station is always available to collect the compressed data and does not address the specific problem of priority-based data preservation in BSN.

Our Main Findings. The *maximum weighted flow* (MWF) problem uncovered in this paper generalizes the classic maximum flow problem [9]. To the extent of our knowledge, this work is the first one to formulate and study the maximum weighted flow problem and design a polynomial algorithm to solve it optimally. Besides, our simulation results show that with 20% of energy cost, our algorithms can preserve 80% of the data items. This result conforms to the well-known Pareto Principle [34], which says that in many societal and economic scenarios, roughly 80% of the outcomes and consequences come from 20% of causes (a.k.a. the "vital few"). Our work complements this classic principle from the networking point of view. In classic network flows [3], the *edge capacity constraint* states that in a flow network, the number of flows on each edge is less than or equal to the edge capacity, and the same flow from source s to t consumes one unit of capacity on all the edges traversed by this flow. Under the general energy model of PDP, however, the relationship between the flow and the edge capacity becomes more complicated. Our model shows that the same flow on different edges could consume different amounts of edge capacities. How this generalized edge capacity constraint affects the existing network flow theories and algorithms is left to future work.

Priority-Based Techniques. In the theory community, researchers have considered edge priorities in maximum flow problems. Kozen [26] studies a lexicographic flow problem wherein edges are assigned priorities, and the goal is to find a lexicographically maximum flow concerning such priority assignment. As stated in [26], a lexicographically max flow is not necessarily a max flow. Another problem, maximum priority flow, is studied in [2, 6]. In this problem, each node specifies a priority ordering for all the edges leaving it, and the flows leaving this node always go through the edges with higher priority before going through the edges with lower priority. Both works do not consider assigning priority to each flow. The most related work to ours is by Fatourou et al. [12]. They study priority-based max-min fairness, wherein each session bears a priority, and the goal is to assign the maximum possible rate corresponding to its priority to each session. However, their focus is the classic theory of max-min fairness, which differs from our goal of maximizing the total weight of all the flows.

A few works study priority-based data dissemination in the sensor network community. Basagni et al. [5] proposed a mathematical model and a distributed heuristic for path finding for an AUV collecting data with decaying value in underwater sensor networks. However, as they did not consider the energy and storage constraints of the sensor nodes, their problem model is very different from ours. Kumar et al. [27] address how to deliver data with different importance (priority) in the

presence of congestion in sensor networks. Kim [25] proposes a quality-of-service MAC protocol based on data priority levels among data transmissions. However, they assume the data is transmitted directly to the base station. In contrast, our unique BSN model and data priorities require new algorithms and solutions for priority-based data preservation.

III. Maximum Weighted Flow Problem (MWF)

In this section, we formulate the maximum weighted flow problem and design an efficient and optimal algorithm.

A. Problem Formulation of MWF

Let $G = (V, E)$ be a flow network. The capacity of an edge $(u, v) \in E$, denoted by $c(u, v)$, is a mapping $c : E \to \mathbb{R}^+$. It represents the maximum amount of flow that can pass through an edge. There are k *source nodes* $S = \{s_1, s_2, ..., s_k\} \subset V$ and one *sink node* $t \in V$. A flow on an edge $(u, v) \in E$, denoted by $f(u, v)$, is a mapping $f : E \to \mathbb{R}^+$, subject to the following two constraints:

1). *Edge Capacity constraint:* $f(u, v) \leq c(u, v), \forall (u, v) \in E$. That is, the flow of an edge cannot exceed its capacity.

2). *Flow conservation constraint:* $\sum_{u \in V} f(u, v) = \sum_{u \in V} f(v, u)$, for each $v \in V \setminus S \cup \{t\}$. The sum of the flows entering a node must equal the sum of the flows exiting a node, except for the source and the sink nodes. For source node $s_i \in S$, the *net flow* is $\sum_{u \in V}(f(s_i, u) - f(u, s_i)) > 0$; for sink node t, the *net flow* is $\sum_{u \in V}(f(t, u) - f(u, t)) < 0$.

Unlike in classic network flow problems (e.g., maximum flow and minimum cost flow), wherein each flow has one unit of weight, in MWF, each net flow out of $s_i \in S$ weights v_i. Therefore, MWF is significantly different from all the existing network flow problems.

Definition 1: (**Total Weight of A Flow.**) Given a flow f in the flow network $G = (V, E)$, its total weight, denoted as $\mathcal{V}_f$, is the sum of weights of all the net flow out of source nodes. That is, $\mathcal{V}_f = \sum_{s_i \in S} \sum_{u \in V} v_i \times (f(s_i, u) - f(u, s_i))$. □

$\mathcal{V}_f$ represents the total weight of flow passing from source nodes to the sink node instead of the total amount of flow desired in the classic maximum flow problem. MWF aims to find a flow f from S to t such that $\mathcal{V}_f$ is maximized. Next, we present an optimal algorithm for MWF called Greedy Optimal Algorithm (GOA).

Greedy Optimal Algorithm (GOA) for MWF. We first transform G into G' by adding a super source node s and a directed edge (s, s_i) with capacity $c(s, s_i) = \infty$ for each $i = 1, 2, ..., k$ and then we apply GOA on G'. GOA (Algorithm 1) is essentially a classic maximum flow augmenting path algorithm, such as Edmonds-Karp algorithm [9], executed in non-ascending order of source nodes' priorities. It first maximizes the amount of flow from s to the source node with the highest priority (from where the flow goes to t) and then updates the residual graph [9] accordingly. If there are still augmenting paths in the residual graph from s to t, it maximizes the flow from s to the source node with the second highest priority. And so on until no more augmenting path can be found from s to t in the residual graph. Since the time complexity of the Edmonds-Karp algorithm is $O(nm^2)$, the running time of the GOA is therefore $O(knm^2)$. There are more efficient maximum flow algorithms, such as Dinitz's blocking flow algorithm [10] with a dynamic tree with $O(mn\lg n)$ running time. However, such algorithms usually rely upon complicated data structures to speed up efficiency and can not be easily implemented.

Algorithm 1: Greedy Optimal Algorithm (GOA) on G'.
Input: G', S, t and v_i, where $s_i \in S$;
Output: flow f and its total weight $\mathcal{V}_f$;
0. **Notations**:
   f: current flow from s to t
   G'_f: residual graph of G' with flow f
1. $f = 0$, $G'_f = G'$;
2. Sort source nodes in non-ascending order of their weights: $v_1 \geq v_2 \geq ... \geq v_k$;
3. **for** $(1 \leq i \leq k)$
4. **while** (G'_f contains an augmenting path *s*-s_i-*t*)
5. Augment flow f along such path;
6. **end while;**
7. **end for;**
8. **RETURN** f and $\mathcal{V}_f$.

Lemma 1: Both GOA and an optimal MWF algorithm yield maximum flow.

Proof: GOA is a maximum flow algorithm with a fixed order of choosing augmenting paths according to source node weights. As each flow reduces one unit of edge capacity in the residual graph, when no more augmenting paths can be found between the source and sink in the residual graph, it yields the maximum amount of flow following the maximum flow-minimum cut theorem [9]. By way of contradiction, if an optimal algorithm is not maximum flow, we can further augment the flow and thus yield a solution with a more significant flow weight. ■

B. Optimality of GOA

Below, we use a constructive approach to prove that GOA is optimal. We first give some notations.

Notations. Let $P = \{P_1, P_2, ..., P_k\}$ be the flow solution yielded by GOA (Algorithm 1), indicating that s_i has P_i amount of net flow in GOA. Let $O = \{O_1, O_2, ..., O_k\}$ be the optimal solution, indicating that s_i has O_i amount of net flow in the optimal solution. WLOG, we assume that $v_1 \geq v_2 \geq ... \geq v_k$.

Let $\Lambda = \{s_{\lambda(1)}, s_{\lambda(2)}, ..., s_{\lambda(a)}\}$, with $1 \leq \lambda(1) < \lambda(2) < ... < \lambda(a) \leq k$, be the set of a source nodes with $P_{\lambda(i)} > O_{\lambda(i)}$. We refer to $P_{\lambda(i)} - O_{\lambda(i)}$ as the *surplus* of $s_{\lambda(i)}$. The total surplus of Λ is $\sum_{i=1}^{a}(P_{\lambda(i)} - O_{\lambda(i)})$.

Let $\Xi = \{s_{\xi(1)}, s_{\xi(2)}, ..., s_{\xi(b)}\}$, with $1 \leq \xi(1) < \xi(2) < ... < \xi(b) \leq k$, be the set of b source nodes with $P_{\xi(j)} < O_{\xi(j)}$. We refer to $O_{\xi(j)} - P_{\xi(j)}$ as the *deficit* of $s_{\xi(j)}$. The total deficit of Ξ is $\sum_{j=1}^{b}(O_{\xi(j)} - P_{\xi(j)})$.

Obviously Λ and Ξ are mutually disjoint, and $a+b \leq k$. According to Lemma 1, the total surplus of Λ equals the total deficit of Ξ, that is, $\sum_{i=1}^{a}(P_{\lambda(i)} - O_{\lambda(i)}) = \sum_{j=1}^{b}(O_{\xi(j)} - P_{\xi(j)})$.

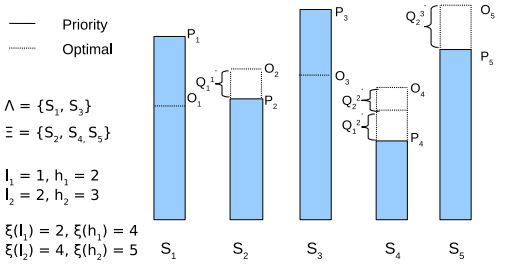


Fig. 2: An example with five source nodes: $s_1, ..., s_5$, illustrating constructive algorithm mapping $\Xi = \{s_2, s_4, s_5\}$ to $\Lambda = \{s_1, s_3\}$. The matching set of s_1 is $\{s_2, s_4\}$, with $P_1 - O_1 = Q_1^1 + Q_1^2$. The matching set of s_3 is $\{s_4, s_5\}$, with $P_3 - O_3 = Q_2^2 + Q_2^3$.

Rewriting $\sum_{j=1}^{b}(O_{\xi(j)} - P_{\xi(j)})$. Next, we are going to rewrite $\sum_{j=1}^{b}(O_{\xi(j)} - P_{\xi(j)})$, a sum of b terms, into a sum of a terms, with i^{th} term having value of $(P_{\lambda(i)} - O_{\lambda(i)})$.

Definition 2: (**Matching Set & Matching Amount.**) The matching set of $s_{\lambda(i)} \in \Lambda$, denoted as $\{s_{\xi(l_i)}, s_{\xi(l_i+1)}, ..., s_{\xi(h_i)}\} \subseteq \Xi$, where $1 \leq l_i \leq h_i \leq b$, is a subset of Ξ whose total matching amount is $(P_{\lambda(i)} - O_{\lambda(i)})$. Here, the matching amount of $s_{\xi(j)}$ to $s_{\lambda(i)}$, denoted by Q_i^j, is the amount of $s_{\xi(j)}$'s deficit that is allocated for $s_{\lambda(i)}$; the total matching amount of $s_{\lambda(i)}$'s matching set is $\sum_{j=l_i}^{h_i} Q_i^j = (P_{\lambda(i)} - O_{\lambda(i)})$. □

In other words, we will rearrange the total deficit of Ξ and map Ξ to Λ, by finding $s_{\lambda(i)}$'s matching set and calculating the corresponding set of matching amount, such that this matching set's total matching amount equals $s_{\lambda(i)}$'s surplus. Algorithm 2 below decides $s_{\lambda(i)}$'s matching set and calculates the matching amount of each element in this matching set.

Algorithm 2: A Constructive Algorithm Mapping Ξ to Λ.
Input: O_i and P_i, $1 \leq i \leq k$.
Output: l_i, h_i, $\{Q_i^{l_i}, Q_i^{l_i+1}, ..., Q_i^{h_i}\}$, $1 \leq i \leq a$.

0 **Notations**:
  $alloc$: total matching amount that is allocated to $s_{\lambda(i)}$;
  $flag = true$ if $l_i = h_i$; $false$ if $l_i < h_i$;
1 $l_1 = 1$;
2 $Q_1^1 = alloc = O_{\xi(1)} - P_{\xi(1)}$;
3 **for** $(1 \leq i \leq a)$
4 $j = l_i$; $flag = true$;
6 **while** $\big(alloc < (P_{\lambda(i)} - O_{\lambda(i)})\big)$
7 $j++$;
8 $Q_i^j = O_{\xi(j)} - P_{\xi(j)}$; $alloc = alloc + Q_i^j$;
10 $flag = false$;
11 **end while;**
12 $h_i = j$;
13 **if** $(flag == true)$ $Q_i^j = P_{\lambda(i)} - O_{\lambda(i)}$;
15 **else**
16 $Q_i^j = (O_{\xi(j)} - P_{\xi(j)}) - \big(alloc - (P_{\lambda(i)} - O_{\lambda(i)})\big)$;
17 **if** $\big(alloc == (P_{\lambda(i)} - O_{\lambda(i)})\big)$
18 $l_{i+1} = j + 1$;
19 $alloc = O_{\xi(l_{i+1})} - P_{\xi(l_{j+1})}$;
20 **else**
21 $l_{i+1} = j$;
22 $alloc = alloc - (P_{\lambda(i)} - O_{\lambda(i)})$;
23 **end for;**
24 **RETURN** $[l_1, l_2, ..., l_a]$, $[h_1, h_2, ..., h_a]$, and Q_i^j.

It is not difficult to check that $(P_{\lambda(i)} - O_{\lambda(i)}) = \sum_{j=l_i}^{h_i} Q_i^j$. Note that $s_{\xi(j)}$ could be in multiple matching sets. That is the deficit of $s_{\xi(j)}$, $O_{\xi(j)} - P_{\xi(j)}$, may be divided into multiple parts, each is allocated to the matching set of a different $s_{\lambda(i)}$. Fig. 2 is an example to illustrate this algorithm. Algorithm 2 immediately gives us the following result.

Lemma 2: $\lambda(i) < \xi(l_i)$, $1 \leq i \leq a$.

Proof: By way of contradiction, assume that there exists j such that $\lambda(j) > \xi(l_j)$, and $\lambda(i) < \xi(l_i)$ for all $1 \leq i \leq j-1$. Since $(P_{\lambda(i)} - O_{\lambda(i)}) = \sum_{j=l_i}^{h_i} Q_i^j$, for all $1 \leq i \leq j-1$, Algorithm 1 found a way to *equalize* number of flows obtained in O to number of flows obtained in P, for all s_i with $i \leq \xi(l_j)$, by moving flows from source nodes in Ξ to source nodes in Λ. Next, when both GOA and optimal algorithm try to find the number of flows for $s_{\xi(l_j)}$, it obtains that $P_{\xi(l_j)} < O_{\xi(l_j)}$, because $s_{\xi(l_j)} \in \Xi$. This contradicts that GOA is a greedy algorithm that finds the maximum number of data to distribute from $s_{\xi(l_j)}$ at this stage. ∎

Theorem 1: GOA is an optimal algorithm. That is, it finds flow with total maximum weight.

Proof: By way of contradiction, assume that GOA is not optimal, therefore $\sum_{i=1}^{k} O_i \times v_i > \sum_{i=1}^{k} P_i \times v_i$. It can be shown that

$$\begin{aligned}
&\sum_{i=1}^{k} P_i \times v_i - \sum_{i=1}^{k} O_i \times v_i \\
= &\sum_{i=1}^{a}(P_{\lambda(i)} - O_{\lambda(i)}) \times v_{\lambda(i)} - \sum_{j=1}^{b}(O_{\xi(j)} - P_{\xi(j)}) \times v_{\xi(j)} \\
= &\sum_{i=1}^{a}(P_{\lambda(i)} - O_{\lambda(i)}) \times v_{\lambda(i)} - \sum_{i=1}^{a}\sum_{j=l_i}^{h_i} Q_i^j \times v_{\xi(j)} \\
= &\sum_{i=1}^{a}\left((P_{\lambda(i)} - O_{\lambda(i)}) \times v_{\lambda(i)} - \sum_{j=l_i}^{h_i} Q_i^j \times v_{\xi(j)}\right).
\end{aligned}$$

According to Lemma 2, for any $1 \leq i \leq a$, $v_{\lambda(i)} \geq v_{\xi(j)}$, the priority of $s_{\xi(j)}$ in $D_{\lambda(i)}$'s matching set. And we have $(P_{\lambda(i)} - O_{\lambda(i)}) = \sum_{j=l_i}^{h_i} Q_i^j$. Each of the above a difference terms is therefore non-negative. Thus we have $\sum_{i=1}^{k} P_i \times v_i \geq \sum_{i=1}^{k} O_i \times v_i$, a contradiction. Note that h_a equals ξ_b due to the fact that $\sum_{i=1}^{k} O_i = \sum_{i=1}^{k} P_i$ (Lemma 1). ∎

IV. Algorithms for PDP under Uniform Energy Model

In this section, we show that the GOA algorithm can solve PDP optimally when applied on a flow network that is appro-

priately transformed from the BSN graph. We first transform the BSN graph $G(V, E)$ to a flow network $G'(V', E')$, as shown in Fig. 3(a). The rationale of this transformation is to convert the storage and energy constraints of sensor nodes in $G(V, E)$ into the edge capacities in the flow network $G'(V', E')$ so that they can play a role in finding the flows of maximum priorities in the data preservation process.

1). 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.
2). Split node $i \in V$ into two nodes: *in-node* i' and *out-node* i''. Add a directed edge (i', i'') with a capacity of E_i, the initial energy level of node i. All the incoming directed edges of node i are incident on i', and all the outgoing directed edges of node i emanate from i''. Therefore the two directed edges (i, j) and (j, i) in Step 1) become (i'', j') and (j'', i').
3). Add a source node s, and connect s to the in-node i' of the DG node $i \in V_d$ with an edge of capacity d_i.
4). Add a sink node t, and connect out-node j'' of the storage node $j \in V_s$ to t with an edge of capacity m_j.

In Step 2) above, the edge (i', i'') is referred to as sensor node i's *internal edge*, and the number of flows on (i', i'') represents the amount of "traffic" that goes through i. $V' = \{s\} \cup \{t\} \cup \{i' : i \in V\} \cup \{i'' : i \in V\}$ and $E' = \{(i'', j') : (i, j) \in E\} \cup \{(j'', i') : (i, j) \in E\} \cup \{(i', i'') : i \in V\} \cup \{(s, i') : i \in V_d\} \cup \{(j'', t) : j \in V_s\}$. We have $|V'| = 2 \cdot |V| + 2$ and $|E'| = 2 \cdot |E| + 2|V|$. Fig. 3(a) shows the flow network transformed from the linear BSN in Fig. 1.

Discussions. The above graph transformation technique is for the following purposes. First, splitting a node into two and adding an edge in between while assigning the initial energy level of the node as the capacity of the newly created edge guarantees that each node cannot exceed its energy capacity. Second, by adding super source node s and super sink node t, and by assigning d_i and m_i to be the capacities of edges connecting s and i' (for source nodes) and i'' to t (for storage nodes), it makes sure that a source node cannot offload more overflow data items than it has and a storage node cannot store more overflow data packets than its storage allows. Note that both source and storage nodes can relay data items of other source nodes.

Theorem 2: GOA algorithm on $G'(V', E')$ is an optimal algorithm for PDP on $G(V, E)$ under the uniform energy model.

Proof: Note that with the above transformation from $G(V, E)$ to $G'(V', E')$, the energy constraints of all sensor nodes and storage capacities of storage nodes are converted to edge capacities in $G'(V', E')$ that need to conform with. Next, we show that the maximum weighted flow from GOA on $G'(V', E')$ corresponds to the data preservation on $G(V, E)$ that preserves the maximum data priorities.

Suppose that GOA gives a net flow of P_i amount from s_i to t. Because GOA is optimal (Theorem 1), $\sum_{i=1}^{k} P_i \times v_i$ is maximum among all the possible flow solutions. With the above transformation from $G(V, E)$ to $G'(V', E')$, P_i is the amount of data items offloaded from source node s_i to sink node t, v_i is the priority of data items generated by s_i. Therefore, the corresponding data offloaded from all s_i to t have the maximum amount of priorities. Solving the maximum weighted flow problem on $G'(V', E')$ provides the optimal solution for the data preservation problem with priority in $G(V, E)$. ∎

EXAMPLE 2: By applying GOA to Fig. 3(a), the final residual graph is shown in Fig. 3(b). It shows that one flow unit comes in and out of the split nodes of source node 1, which means it successfully offloads one of its data items. Fig. 3(c) shows one of the final residual graphs by applying classic maximum flow algorithms (e.g., Edmonds-Karp [9]), which does not consider the priorities of the flows. □

However, Fig. 3(b) and Fig. 3(c) show that both GOA and class maximum flow yield the same amount of flow (i.e., one). This is confirmed by Lemma 1 in Section III. We have the following observation of the optimal solution for PDP and classic maximum flow.

Observation 1: In any PDP optimal solution under the uniform energy model, a storage node does not relay data until its storage capacity is full. However, a DG node could relay data items for other DGs even if it has not finished offloading all its own data items. Nonetheless, a maximum flow solution always exists when a DG node relays data items only after it has finished distributing all its own data items. □

A consequence of Observation 1 is that data is always offloaded to the nearest storage node with available storage before being offloaded further in the optimal algorithm. We call this the *cascading effect* of data preservation, which resembles a waterfall that descends by flowing on a higher level of rocks before flowing on a lower level. This effect makes our data preservation algorithms amenable to localized distributed implementation, which will be presented in Section VI. However, a DG node could relay data items for other DGs even if it has not finished distributing all its own data items, as node 2 in Fig. 1.

A Heuristic Algorithm for PDP. Next, we present an efficient heuristic algorithm (Algorithm 3) for PDP. Unlike the GOA, it does not use the flow network $G'(V', E')$ and its residual graphs to compute the data preservation solution. Instead, it is directly applied on the BSN graph $G(V, E)$. Like the GOA, this algorithm offloads data in the descending order of their priorities. In each iteration, a DG with the highest data priorities offloads its data items to the closest storage node with available storage as long as all the involved nodes have enough energy (i.e., all the nodes along the path from this DG to this storage node have at least one unit of energy). Algorithm 3 stops when no more data can be offloaded due to network partitions caused by the energy depletion of some sensor nodes.

Algorithm 3: Heuristic Algorithm on $G(V, E)$.
Input: A BSN graph $G(V, E)$.
Output: flow f and its total weight $\mathcal{V}_f$.

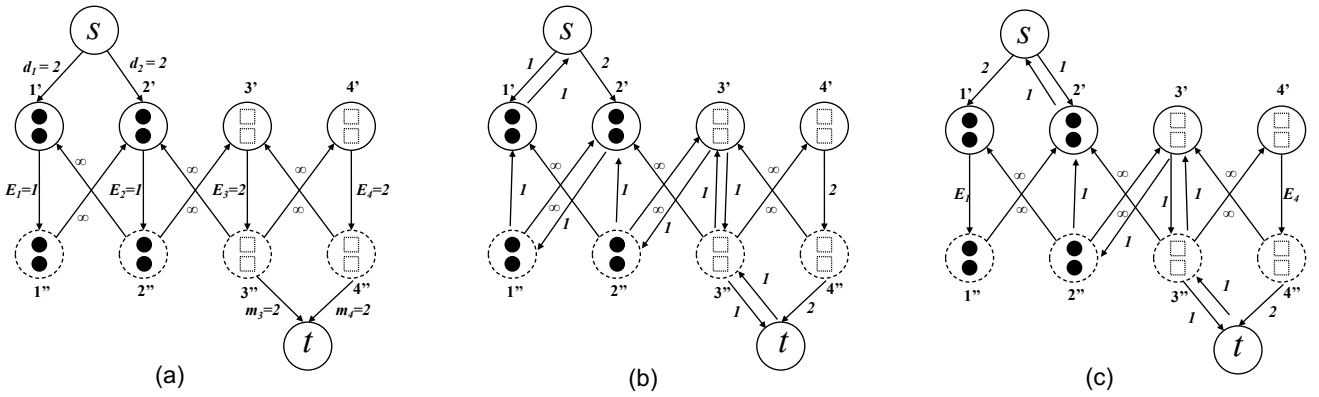


Fig. 3: (a) Flow network $G'(V', E')$ (i.e., the initial residual graph) transformed from linear BSN graph in Fig. 1. (b) The final residual graph obtained by applying GOA, showing node 1 offloads one of its data items with a priority of 3. (c) One of the final residual graphs obtained by applying classic maximum flow algorithms (e.g., Edmonds-Karp [9]), showing node 2 offloads one of its data items with a priority of 1.

1. Sort all the DGs in descending order of their priorities: $v_1 \geq v_2 \geq ... \geq v_k$
2. **for** $(1 \leq i \leq k)$
3. **while** (It can still distribute a data item from DG i
4. to a storage node)
5. Distribute it to the closest storage node;
6. Update the energy levels of all involved nodes;
7. Update f and $\mathcal{V}_f$;
8. **end while;**
9. **end for;**
10. **RETURN** f and $\mathcal{V}_f$.

Time Complexity. Due to space constraints, we omit detailed analysis here. Its time complexity is $O(km + k\bar{d}n)$, where $\bar{d}$ is the average number of data items of each DG. This is more efficient than GOA, which is $O(knm^2)$.

V. Algorithms for PDP under General Energy Model

In this section, we first present an ILP formulation for PDP, and then study the feasible conditions for PDP.

A. ILP for PDP

We present an ILP formulation viz. ILP(A) for the PDP under the general energy model is as follows. Here, the decision variable x_{ij} indicates the number of flows on edge $(i, j) \in E'$ in $G'(V', E')$.

$$(A) \quad \max \sum_{i \in V_d} (x_{s,i'} \cdot v_i) \tag{1}$$

s.t.

$$x_{s,i'} \leq d_i, \quad \forall i \in V_d \tag{2}$$

$$x_{i'',t} \leq m_i, \quad \forall i \in V_s \tag{3}$$

$$x_{s,i'} + \sum_{j:(i,j)\in E} x_{j'',i'} = x_{i',i''} = \sum_{j:(i,j)\in E} x_{i'',j'}, \quad \forall i \in V_d \tag{4}$$

$$\sum_{j:(i,j)\in E} x_{j'',i'} = x_{i',i''} = \sum_{j:(i,j)\in E} x_{i'',j'} + x_{i'',t}, \quad \forall i \in V_s \tag{5}$$

$$E_i^r \times \sum_{j:(i,j)\in E} x_{j'',i'} + \sum_{j:(i,j)\in E} \left(E_i^t(j) \times x_{i'',j'}\right) \leq E_i \quad \forall i \in V \tag{6}$$

Like the GOA for the uniform energy model, the ILP is applied on the flow network $G'(V', E')$ shown in Fig. 3(a). In ILP (A), Objective (1) is to find the maximum data priorities that can be offloaded from DGs to storage nodes in the entire BSN. Inequality (2) indicates the maximum number of data items DG_i can offload is d_i, the initial number of data items DG_i has. Inequality (3) indicates the maximum number of items storage node i can store is m_i, the storage capacity of storage node i. Equation (4) shows the flow conservation for DGs, where the number of its own data items offloaded plus the number of data items it relays for other DGs equals the number of data items it transmits. Equation (5) is the flow conservation for storage nodes, which says that data items received by storage node i are either relayed to other nodes or stored at i. In both (4) and (5), $x_{i',i''}$ represents the amount of "traffic" that goes through i. Recall (i', i'') is sensor node i's internal edge and the number of flows on (i', i''), $x_{i',i''}$, represents the number of flows that goes through i. For source node i, such flows include its own data items and items received from other nodes. For storage node i, such flows include data items received from others, some transmitted to others while others are stored in its local storage. Inequality (6) specifies the energy constraints of sensor nodes (including DGs and storage nodes), wherein the energy consumption on a sensor node i cannot exceed its initial energy level E_i.

Discussions. In classic network flows, each flow unit *consumes* one unit of flow capacity; with one more flow going through an edge, the available capacity on this edge will decrease by one.

As such, existing network flow literature [3] generally refers to the *edge capacity constraint* as the amount of flow on an edge must be less than or equal to the edge's capacity. This is also the case for maximum weighted flow formulation in Section III, which corresponds to PDP under a uniform energy model. In ILP(A), however, for the internal edge (i', i''), the relationship between its flows $x_{i',i''}$ and its capacity E_i is much more complicated, as demonstrated in Equations (4) and (5) and Inequality (6). In particular, one flow unit on different edges could consume different amounts of edge capacities. We refer to it as *generalized edge capacity constraint* and will study its effects on the existing network flow theories and algorithms in the future.

EXAMPLE 3: We introduce a simplified version of the general energy model wherein for each sensor node, sending or receiving a data item each costs 0.5 units of its energy. That is, when a DG sends a data item, it costs 0.5 unit of its energy; when a node (a DG node or storage node) relays (receives and then sends) a data item, it costs 1 unit of energy; when a storage node finally receives and stores a data item, it costs 0.5 unit of energy. Under this model, the energy consumption of sending a data item from a DG to a storage node equals the number of edges it traverses. The optimal solution for Fig. 1 would be offloading one data item of node 1 to node 3, resulting in a total preserved priority of 3. □

However, the maximum flow in Fig. 1 under this model would be two, offloading two data items of node 2 to node 3, giving total preserved priority of 2. Therefore, we give the below theorem without proof.

Theorem 3: Under general energy model, the maximum weighted flow is not necessarily a maximum flow, and a maximum flow is not necessarily a maximum weighted flow.

Note that this result is very different from Lemma 1 in Section III, which says that under the uniform energy model, the maximum weighted flow is still a maximum flow. Next, we answer an important and related question: Under which conditions is PDP *feasible*; that is, only some of the overflow data items can be offloaded? Note that PDP assumes that only some overflow data items can be preserved due to energy and storage insufficiency of the sensor nodes (otherwise, PDP is a trivial problem as offloading all the data items achieves maximum preserved priority).

B. Feasible Conditions for PDP Instances

Given any *BSN instance*, which includes the network topology, source nodes with their overflow data items and data priorities, and storage nodes with storage capacities, we are interested in finding if it leads to the PDP wherein not all the overflow data items can be preserved. If so, this instance is called an *PDP instance*. Note that PDP instances could be caused by either insufficient storage spaces or insufficient energy power of sensor nodes.[2] We can find if a BSN instance is a PDP instance as follows.

[2]In contrast, existing work [39, 41] only assumes insufficient storage spaces in the BSN (i.e., $\sum_{i=k+1}^{n} m_i < q$) and applies data aggregation techniques to reduce the sizes of the overflow data packet before offloading them.

First, we formulate another ILP viz. ILP(B) as below. We omit its constraints as they are the same as those of ILP(A).

$$(B) \qquad \max \sum_{i \in V_d} x_{s,i'}. \tag{7}$$

Eq. 7 is to maximize the total amount of flow out of the source nodes V_d; i.e., the total number of preserved data items. Next, we show the feasible conditions for a PDP instance.

Theorem 4: Given any BSN instance $G(V, E)$ under the general energy model, if $\sum_{i \in V_d} x_{s,i'}$ computed in ILP(B) is less than $q = \sum_{i=1}^{k} d_i$, the total number of overflow data items in the BSN, then this BSN instance is a PDP instance.

Proof: First, when $\sum_{i \in V_s} x_{si'} < q$, there are less than q amount of flows going from s to t in $G'(V', E')$. As the edge capacity between s and i' is d_i, it must be that at least exists one i, $1 \leq i \leq k$, that that there are less than d_i amount of flows going from s into i'. Due to flow conservation in all nodes $V' - \{s, t\}$, there must be less than d_i amount of flows going out of i''. Meanwhile, the initial energy level of node i (i.e., E_i) is now the capacity of edge $(i', i'') \in E''$. As the energy consumption of any node i, which is represented by the l.h.s of Inequality (8) and (9) in ILP (A), is less than E_i, it guarantees that node i does not exceed its energy capacity during data preservation in $G(V, E)$.

Therefore, all the d packets in BSN $G(V, E)$ can be offloaded from their data nodes to storage nodes while satisfying the energy constraints of sensor nodes, yielding feasible data preservation. ∎

Discussions. That is, in ILP(B), when the total flow $\sum_{i \in V_d} x_{s,i'} < q$, we will then apply ILP(A) to solve the PDP problem. ILP(B) is the ILP for the classic maximum flow problem, with some twist. Recall maximum flow is to find the maximum number of flows that can be transmitted in a flow network under the edge constraints. In maximum flow, each flow costs one unit of the edge capacity. However, in ILP(B), each flow (i.e., each data item offloading) could cost different amounts of energy on different nodes (due to the general energy model), thus costing different units of edge capacity in the flow network $G''(V'', E'')$.

VI. Distributed Algorithms for PDP

We design a distributed algorithm by combining the idea behind the push-relabel maximum flow algorithm [16] and data priority-based data preservation. In the push-relabel algorithm, nodes only need to know their neighbors and the capacities of the edges with their neighbors by sending and receiving messages. Therefore, it is desirable for a distributed sensor network environment.

Overview of Push-Relabel Algorithm [16]. Given a flow network $G(V, E)$ with $|V| = n$ and $|E| = m$, a source node s and a sink node t, the push-relabel algorithm works as follows. It begins by sending as much flow out of s as allowed by the capacities of the edges coming out of s. Then, at each

iteration, it considers a node with more incoming flow than outgoing flow (the difference between them is called *excess flow* of the node). The node then routes the excess flow to their neighbors, and so on. This operation is called $push$. To make progress, the algorithm defines a height function $h : V \to N$ and initially, $h(s) = n$; $h(t) = 0$; and $h(u) = 0$ for all $u \notin V \setminus \{s, t\}$. This is based on the intuition that the flow always goes "downhill", and node u sends the extra flow to a neighbor v only if $h(u) > h(v)$. When a node can no longer send out its excess flow, it increases its height and pushes the flow to the node with a lower height than it. This operation is called $relabel$. A maximum flow is obtained when no more overflowing nodes are left except the sink and possibly the source node. The running time of the push-relabel algorithm is $O(n^2m)$. The detailed operations of the push-relabel algorithm on a node u are presented in Algorithm 4.

Algorithm 4: Push-Relabel (u)

0. **Notations:**
   $e(u)$: node u's excess flow
   $h(u)$: node u's height
   $cap(u, w)$: residual capacity of (u, w)
1. **if** $e(u) > 0$
2. **while** $\big(e(u) > 0$, there exists (u, w) s.t.
3. $h(u) = h(w) + 1$, and $cap(u, w) > 0\big)$
4. Push $y = \min\{e(u), cap(u, w)\}$ through
5. (u, w) by sending a message to w;
6. $e(u) = e(u) - y$; $e(w) = e(w) + y$;
7. update $cap(u, w)$;
8. **end while;**
9. **if** $e(u) > 0$
10. $h(u) = 1 + \min\{h(w) : cap(u, w) > 0\}$;
11. Broadcast $h(u)$ to neighboring nodes;
12. **end if;**
13. **end if;**
15. **RETURN**

Distributed Data Preservation with Data Priority. The flow in the push-relabel algorithm bears intrinsic resemblance with the cascading effects exhibited by the data flow in our data preservation problem (see Observation 1). In push-relabel, the flow goes through the network as water flows through downhills. In data preservation, data is always offloaded to the nearest storage before being offloaded farther away. Therefore, the push-relabel algorithm is particularly suitable for our data preservation scheme.

However, a few modifications to the push-relabel algorithm are needed to make the distributed data preservation work. First, since the push-relabel algorithm finds maximum flow while data preservation finds maximum preserved priorities, the DGs need to coordinate with each other so that DGs with higher priorities push data into the network before DGs with lower priority do. Second, the energy constraint of each node should be represented in the flow network, and the energy consumption of sending and receiving packets by each node should be taken into account. To handle energy constraints and energy consumptions of nodes, nodes are split according to Section IV. Third, the push-relabel algorithm determines the maximum flow from a single source to a single sink. In contrast, in a sensor network, there are multiple data-generating sensor nodes and multiple sensor nodes collecting and storing sensed data. Therefore, as specified in Section IV, a virtual source node s is connected with the in-node of each DG, with the edge capacity as the number of data items of each DG, and a virtual sink node t is connected with the out-node of each storage node, with its storage capacity as edge capacity.

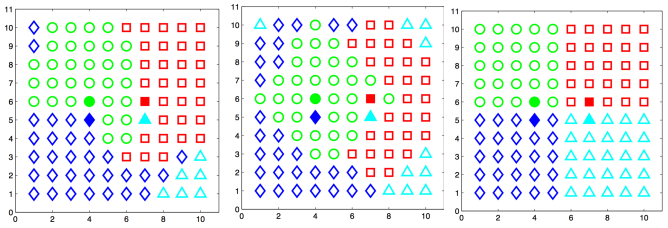
The distributed data preservation begins with this virtual source pushing the maximum allowable number of flows to the in-node of the DG with the highest priority, which continues the push-relabel process until no nodes with the excess flow (except virtual source and sink) exist. Then, the virtual source pushes to the in-node of the DG with the second highest priority. The algorithm works in rounds; in each round, a node with positive excess performs push-relabel. Such synchronization prevents multiple nodes from sending their packets to their neighbors simultaneously. When there are multiple neighbors with the same number of heights, one of them is randomly picked. The distributed algorithm stops when the DG with the lowest priority finishes the push-relabel process or no more flows can be pushed into the network. The detailed data preservation operations are presented in Algorithm 5.

Algorithm 5: Distributed Data Preservation on $G'(V', E')$

1 Each DG broadcasts its priority to the network;
2. **for** $\big($Each DG in the descending order of its priority$\big)$
3. s pushes maximum allowable data to this DG;
4. **while** $\big($there exists a node u with positive excess$\big)$
5. Push-Relabel(u);
6. **end while;**
7. **end for;**
8. **RETURN**

Theorem 5: The distributed data preservation algorithm preserves maximum total priorities. It runs in $O(kn^2)$ time and uses $O(n^2m)$ messages.

Proof: The distributed algorithm is just a distributed implementation of the optimal GOA algorithm. Therefore, it preserves maximum total priorities due to the optimality of the GOA algorithm (Theorem 2). The time and message optimality of the distributed data preservation algorithm is due to the time and message optimality of the distributed push-relabel algorithm [16]. It is shown in [16] that the distributed push-relabel algorithm in a graph $G'(V', E')$ runs $O(|V'|^2)$ in time and uses $O(|V'|^2|E'|)$ messages. Since $|V'| = 2n + 2$ and $|E'| = 2m + 2n$ (Section IV), its time and message complexities are $O(n^2)$ and $O(n^2m)$ respectively. The distributed data preservation differs from the push-relabel algorithm in lines 1 and 2 in Algorithm 5. Line 1 incurs $O(kn) = O(n^2)$ broadcast messages, and Line 2 increases the running time by at most



(a) Optimal. (b) Heuristic. (c) Edmonds-Karp.

Fig. 4: Data preservation in a grid of 10×10 obstructed by insufficient storage. There are four DGs, which are solid circle (with data priority 8), solid square (with data priority 6), solid diamond (with data priority 4), and solid triangle (with data priority 2). Each DG has 30 data items. The remaining 96 nodes are storage nodes; each has one storage and can store one data item. Each node has an initial energy level of 30 units. The preserved data items are corresponding hollow shapes. The number of preserved data items (from highest priority to lowest) are (a) $[30, 30, 30, 6]$ in Optimal, (b) $[30, 28, 28, 10]$ in Heuristic, and (c) $[24, 24, 24, 24]$ in Edmonds-Karp.

k times. Therefore the distributed algorithm runs in $O(kn^2)$ and uses $O(n^2m + n^2) = O(n^2m)$ messages. ∎

VII. Performance Evaluation

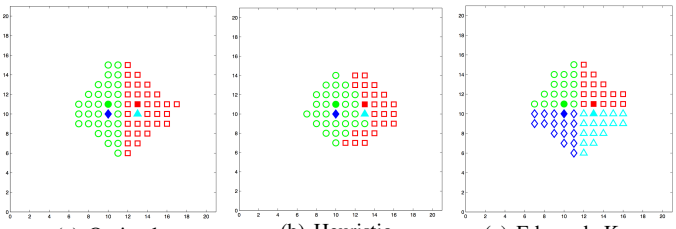
In this section, we compare our priority-based data preservation algorithms with data preservation algorithms based on classic maximum flow, including Edmonds-Karp (for uniform energy model), ILP(B) (for general energy model), and push-relabel (for distributed algorithm). We assume all the sensor nodes have the same initial energy level for both energy models. Table II lists all the compared algorithms. We first investigate the uniform energy model in Section VII-A and then consider the general energy model in Section VII-B. Where applicable, each data point averages over 20 PDP instances, and the error bars indicate a 95% confidence interval. For each PDP instance, we apply all the compared algorithms on the same PDP instance for a fair comparison. We write our simulators in Java on a MacBook Pro with a 3.68 GHz 12-core Apple M2 Max processor and 32 GB RAM.

A. Uniform Energy Model.

We compare the performance of the optimal GOA algorithm (referred to as **Optimal**), the greedy heuristic (referred to as **Heuristic**), and the distributed algorithm (referred to as **Distributed**). We also implement the Edmonds-Karp maximum flow algorithm [9] (referred to as **Edmonds-Karp**), which does not consider flow priorities when preserving data.

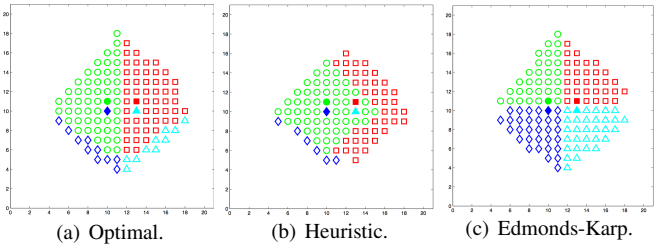
TABLE II: Summary of Compared Algorithms.

	Uniform Model	General Model	Maximum Flow
Centralized	GOA (Algo. 1), Greedy (Algo. 3)	ILP(A), Greedy (Algo. 3)	Edmonds-Karp, ILP(B)
Distributed	Distributed PDP (Algo. 5)	Distributed PDP (Algo. 5)	Push Relabel [16]



(a) Optimal. (b) Heuristic. (c) Edmonds-Karp.

Fig. 5: Data preservation blocked by insufficient energy at $E_i = 5$.



(a) Optimal. (b) Heuristic. (c) Edmonds-Karp.

Fig. 6: Data preservation blocked by insufficient energy at $E_i = 10$.

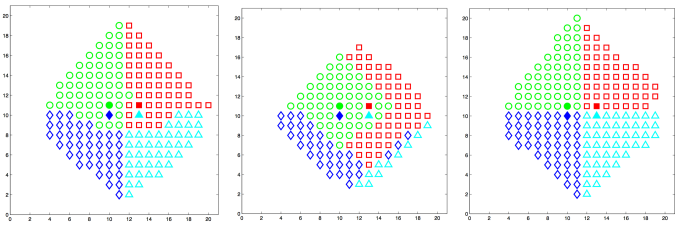
Edmonds-Karp takes place in iterations; each iteration finds the shortest augmenting path in the residual graph of the flow network shown in Fig. 3(a). Although there are many implementations of maximum flow algorithms [3], as Edmonds-Karp takes the shortest path in each round, it is the most energy-efficient.

Visual Performance Comparison in Grid Networks. We adopt grid topologies only for clear visual comparison. We investigate two scenarios in the grid network that obstruct data preservation. There is not enough space in the network to store all the overflow data (i.e., $\sum_{i=k+1}^{n} m_i < q$), or sensor nodes do not have enough energy to preserve all the overflow data (i.e., $\sum_{i=k+1}^{n} m_i > q$).

Data Preservation Obstructed by Insufficient Storage. In this case, we set the grid network size as 10×10, as shown in Fig. 4. There are four DGs represented using four different shapes: solid circle (with data priority 8), solid square (with data priority 6), solid diamond (with data priority 4), and solid triangle (with data priority 2). These four DGs are located at (4, 6), (7, 6), (4, 5), and (7, 5), respectively. Each DG has 30 data items to be preserved. The preserved data items (i.e., offloaded into the grid) are represented using the corresponding hollow shapes. Each storage node has one storage and can store one data item. We set the initial energy level of each node as 30 units. Fig. 4 shows the number of data items (from highest priority to lowest) offloaded by Optimal

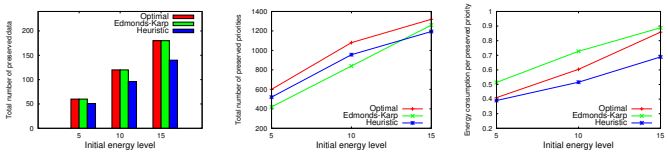
TABLE III: Quantitative results of Fig. 4.

	Optimal	Heuristic	Edmonds-Karp
Number of Preserved Data	96	96	96
Total Preserved Priority	552	540	480



(a) Optimal. (b) Heuristic. (c) Edmonds-Karp.

Fig. 7: Data preservation blocked by insufficient energy at $E_i = 15$.



(a) Total preserved data. (b) Total preserved priority. (c) Energy per priority.

Fig. 8: Quantitative analysis of results in Figs. 5, 6, and 7.

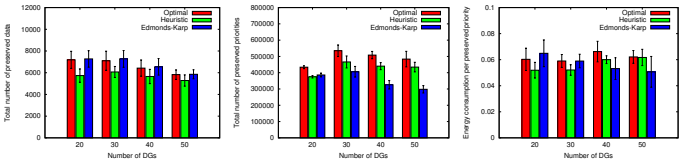
is $[30, 30, 30, 6]$, offloaded by Heuristic is $[30, 28, 28, 10]$, and offloaded by Edmonds-Karp is $[24, 24, 24, 24]$. It shows that Optimal and Heuristic try to preserve data with higher priority, while Optimal does much better to "filter" out the low-priority data. Edmonds-Karp, however, offloads the same amount of data for different priorities as it is "priority-blind". Table III summarizes the comparison results from Fig. 4. It shows that Optimal has the most significant total preserved priority, which Heuristic follows, and Edmonds-Karp gives the smallest total preserved priority. However, they all preserve the same amount of data items constrained by insufficient storage.

Data Preservation Blocked by Insufficient Energy. We also investigate data preservation when the network has insufficient energy to preserve all the overflow data. We set the network size as 20×20, and the four DG are located at (10, 10), (13, 10), (10, 11), and (13, 11), respectively. Each DG has 50 data items to preserve. We visualize the preservation results in Figs. 5, 6, and 7 by increasing the initial energy level of sensor nodes E_i from 5 to 10 to 15, respectively. They show that with the increase of E_i, the number of preserved data items is increased for all three algorithms. Taking a closer look, however, we observe that Optimal and Heuristic only preserve the two data types of higher priorities (circles and squares) at a lower energy level of $E_i = 5$, and begin to preserve the other two types of lower priorities (diamonds and triangles) at intermediate energy level of $E_i = 10$, and finally preserve relatively large amount of the two low-priority data types at higher energy level of $E_i = 15$. All these demonstrate these two algorithms' priority-preserving efforts. In contrast, the Edmonds-Karp always equally preserves all four data types at different energy levels, aiming to maximize the number of preserved data items.

Fig. 8 shows the quantitative analysis of the results in Figs. 5, 6, and 7. Fig. 8(a) shows both Optimal and Edmonds-Karp preserve the largest and equal amount of data items, validating our theoretical results that under the uniform energy model, MWF is indeed an MF. However, Fig. 8(b) shows that the Optimal yields the largest total preserved priority among the three algorithms. As energy efficiency is also a concern in data preservation, we put forward a metric of *energy per priority*, which is the ratio of the total energy consumption in the data preservation versus the total preserved data priorities, as shown in Fig. 8(c). Heuristics achieves the lowest energy per priority ratio across different energy levels, performing better than the Optimal and Edmonds-Karp. This demonstrates that Heuristic, a competitive data preservation algorithm, can balance preserved priorities and energy costs incurred. Finally, it shows that Optimal gives a lower energy per priority ratio than Edmonds-Karp does, showing that It preserves the largest amount of priority and is an energy-efficient data preservation algorithm.

Performance Comparison in General Topologies. Next, we consider the general topologies of the BSN, wherein 100 sensor nodes are randomly generated in a field of $2000\text{m} \times 2000\text{m}$ and the transmission range as 200m. The initial energy of nodes is 500 mJ. Each DG has 500 data items, and each is 400 bytes. The storage capacity of each storage is 51.2 Kbytes. The priority of each DG is a random number in [1, 100]. Fig. 9 compares the performances of Optimal, Heuristic, and Edmonds-Karp, by varying numbers of DGs in [20, 30, 40, 50]. Fig. 9 (a) shows that Edmonds-Karp and Optimal preserve the same amount of data, more than Heuristic, again validating that under the uniform energy model, MWF is indeed a maximum flow. Even so, Fig. 9 (b) shows that Optimal preserved more priorities than Heuristic, which preserves more than Edmonds-Karp. This is because Optimal maximizes the total preserved priority while the other two do not. This seems more prominent when DGs are large (at 30, 40, and 50). However, Fig. 9 (c) shows Edmonds-Karp yields less energy consumption per preserved priority than Optimal and Heuristic most of the time. This is due to the algorithm, which always finds the global minimum energy path in the residual graph when offloading a data item in each round.

Comparing Priority-Based Distributed Algorithm and Distributed Push-Relabel Algorithm. Fig. 10 compares distributed data preservation with data priorities (referred to as Distributed) and without considering data priorities (referred to



(a) Total preserved data. (b) Total preserved priority. (c) Energy per priority.

Fig. 9: Performance Comparison in General BSN Topologies.

as PushRelabel). We use and modify the implementation of a distributed push-relabel algorithm available at [1]. It shows that both algorithms achieve the same amount of distributed data. However, the Distributed yields a higher total preserved priority than PushRelabel. Due to the broadcast messages coordinating the data distribution of different DGs, distributed data preservation incurs more energy consumption.

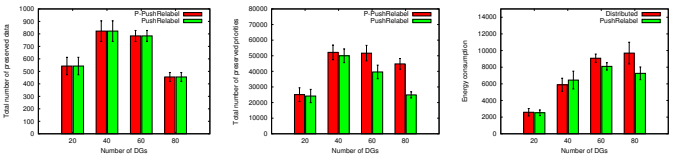
B. General Energy Model.

Finally, we compare the three algorithms under the general energy model. We refer to the ILP(A) as **MWF** (i.e., maximum weighted flow), ILP(B) as **MF** (i.e., maximum flow), and the Algo. 3 as **Greedy**. We don't use grid networks as the general energy model degenerates to the uniform energy model in a grid (with all its edges having the same unit weight). For each BSN instance, 100 sensor nodes (with a transmission range of 200m) are randomly generated in a field of 2000m $\times$ 2000m; 50 are DGs, and 50 are storage nodes. Unless otherwise mentioned, $d_i = m_i = 100$; each DG has 100 data items (each 400 bytes), and each storage can store 100 overflow data packets. In this case, the BSN will be entirely stored if all the 5000 data items are preserved. The priority of each DG is a random number in [1, 100]. Given any BSN instance from a randomly generated BSN graph, we first use ILP(B) to find the minimum E_i such that all the data items can be offloaded successfully. We refer to this energy level as the *minimum feasible energy* (MFE) for this BSN instance.

Energy Insufficiency. We first study data preservation due to the insufficient energy powers of sensor nodes. Fig. 11 shows the performance comparison by varying the sensor nodes' initial energy levels E_i as 20%, 40%, 60%, and 100% of the MFE value for that BSN instance. Fig. 11(a) shows that with the increase of E_i, the total preserved priority for all the algorithms increases. It also shows both MWF and Greedy achieve more preserved priority than MW, while MWF has the most. This demonstrates that MW is not necessarily a MWF. In particular, MWF outperforms MF by 12.13% at $E_i = 20\%$ of MFE. Their performance difference (i.e., the difference of their preserved priorities divided by MW's preserved priority), shown in Table IV, decreases with the increase of E_i. This is because when increasing E_i, the data-preserving capabilities of both algorithms will converge by preserving more and more data items and finally preserving all the available priority when all the data items are preserved at $E_i = 100\%$ of MFE. Fig. 11(b) shows that at $E_i = 20\%$ of MFE, both MWF and MF preserve more than 4000 out of the total 5000 (i.e., 80% of) data items, while it takes another 80% of MFE for them further to preserve the rest of 20% of the data items. This result conforms to the well-known Pareto Principle [34], which says that in many societal and economic scenarios, roughly 80% of the outcomes and consequences come from 20% of causes (a.k.a. the "vital few"). While the Pareto Principle was observed mainly in societal and economic settings, our work complements it by validating it from the perspective of resource allocation in computer networking.

Besides, Fig. 11(b) shows that MF preserves more data items than MWF and Greedy, showing that MWF does not necessarily achieve the maximum amount of flow. At $E_i = 100\%$ of MFE, it can be seen both MF and MWF can successfully preserve all the 5000 data items while Greedy cannot. This can be explained by Fig. 11(c), which shows Greedy is the least energy-efficient among the three algorithms. MF's underlying implementation of Edmonds-Karp algorithms always looks for the shortest path (i.e., the most energy-efficient) to preserve the data items, so it is the most energy-efficient. To investigate the tradeoff between preserved priority and incurred cost, we calculate the *priority-cost ratio* (i.e., the ratio of the total preserved priority and total energy cost) achieved by each algorithm, as shown in Fig. 11(d). This ratio decreases with the increase of E_i for all algorithms, while MF gives a larger priority-cost ratio than the other two. With less energy, Greedy and MWF can choose higher priority packets over lower priority packets, resulting in higher value-cost ratios. With more energy, all algorithms can preserve more packets regardless of their priority values, thus resulting in lower value-cost ratios.

Storage and Energy Insufficiency. Finally, we investigate data preservation due to the insufficiency of both energy and storage. Fig. 12 increases the number of data items d_i at each DG from 50, 60, ..., to 100, while keeping $E_i = 20\%$ of MFE and $m_i = 50$ for all the sensor nodes; therefore, both energy and storage are not enough to preserve all the data items in all the cases. Although Fig. 12(a) and Fig. 12(b) show that preserved priority and preserved data items increase when increasing d_i, there are a few different and non-intuitive observations compared to Fig. 11. First, the performance differences between MWF and MF are much more significant, between 16.78% and 33.29%, as shown in Table IV. This shows that MWF is superior to MW in more resource-scarce scenarios, where MWF can still utilize minimal energy and storage resources to preserve the highest-priority data items.

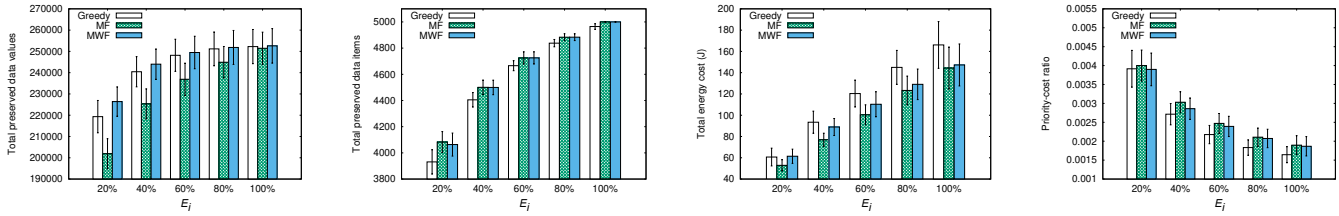


(a) Total preserved data. (b) Total preserved priority. (c) Total energy cost.

Fig. 10: Comparing Distributed and PushRelabel.

TABLE IV: Performance difference between MWF and MF in preserved data priorities by varying E_i and d_i. Here, the performance difference $= \frac{\text{MWF}-\text{MF}}{\text{MF}}$; (i.e., the difference of their preserved priorities divided by MF's preserved priority).

Varying E_i (MFE %), $d_i = m_i = 100$	20	40	60	80	100
	12.13	8.28	5.31	2.84	0.45
Varying d_i, $m_i = 50$, $E_i = 20\%$ MFE	60	70	80	90	100
	16.78	22.63	27.35	31.08	33.29



(a) Total preserved priority. (b) Total preserved data items. (c) Total energy cost. (d) Priority-cost ratio.

Fig. 11: Varying initial energy level E_i (in percentage of MFE) under general energy model. Here, $d_i = m_i = 100$.

Second, with the increase of d_i, the performance differences between MWF and MF increase, whereas they decrease with the increase of E_i in Fig. 11. A close look at Fig. 12(a) shows that with the increase of d_i, the increase of preserved priorities for MWF is much more significant than that for MW. Despite the insufficiency of storage and energy, increasing d_i allows MWF to discard low-priority data for high-priority data, thus increasing its preserved priorities. In contrast, as MW is blind towards the data priority, increasing d_i can only increase preserved priorities as much as storage and energy allow. Fig. 12(c) shows that although the total energy cost of data preservation increases for MWF and Greedy, MW has no clear trend. This might be due to the randomness of how the ILP implementation finds the best flow arcs to maximize throughput without considering the value and cost. Finally, Fig. 12(d) shows that with the increase of d_i, the priority-cost ratios slightly increase for all the algorithms while MF has the largest ratios, striking a good balance between preserved priorities and energy cost.

VIII. Conclusion and Future Work

The maximum weighted flow problem studied in this paper is uniquely derived from the priority-based data preservation problem in base station-less sensor networks of many emerging sensing applications. It is a theoretically fundamental problem since it generalizes the classic maximum flow problem. Because of this theoretical root, the techniques proposed in this paper could be applicable in any network applications in which data priorities and resource constraints coexist, such as flow control in data centers and crowd-sourcing in smartphone communication. It is a very simple but general information producer and consumer model that has yet to be adequately explored in any other context.

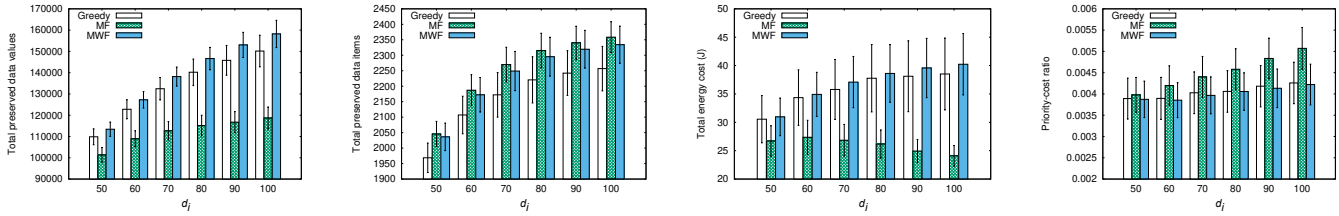
This work establishes an architectural framework with several potential future directions. Currently, the PDP is a static problem in which the data to be preserved is generated at the beginning and only once. In future work, we will address a real-time problem where data is generated and transmitted dynamically and periodically, and storage-depleted nodes may vary over time. Second, the paper assumes that the overflow data of DGs can be offloaded to only the storage nodes. To maximize the total priorities preserved, it would be interesting to explore if certain DGs of low priority can discard their locally generated data and make room to store the data from other DGs of high priority. Third, we will study how *generalized edge capacity constraint* derived from the general energy model, wherein one unit of flow from source s to sink t could cost different amounts of edge capacities on different edges, will affect the existing network flow theories and algorithms. Finally, we will consider the spatial correlation among the data items. Designing energy-efficient data preservation techniques that take advantage of spatial correlation to reduce data redundancy while maximizing their priorities becomes a new and exciting problem.

Acknowledgment

This work was supported in part by NSF Grants CNS-1419952 and CNS-2131309.

References

[1] http://avglab.com/andrew/soft.html.
[2] Maximum priority flow. http://www.nada.kth.se/ viggo/wwwcompendium/node120.html.
[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
[4] S. Basagni, L. Boloni, P. Gjanci, C. Petrioli, C. A. Phillips, and D. Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *Proc. of INFOCOM*, 2014.
[5] S. Basagni, L. Bölöni, P. Gjanci, C. Petrioli, C. A. Phillips, and D. Turgut. Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 988–996, 2014.
[6] Mihir Bellare. Interactive proofs and approximation: Reductions from two provers in one round. In *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, pages 266–274, 1992.
[7] Sobin CC, V. Raychoudhury, G. Marfia, and A. Singla. A survey of routing and data dissemination in delay tolerant networks. *Journal of Network and Computer Applications*, 67:128–146, 2016.
[8] C. T. Chou, A. Ignjatovic, and W. Hu. Efficient computation of robust average of compressive sensing data in wireless sensor networks in the presence of sensor faults. *IEEE Transactions on Parallel and Distributed Systems*, 24(8):1525–1534, 2013.
[9] Thomas Corman, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.
[10] Yefim Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. page 1277–1280, 1970.
[11] L. Terray et al. From sensor to cloud: An iot network of radon outdoor probes to monitor active volcanoes. *Sensors*, 20(10), 2020.
[12] Panagiota Fatourou, Marios Mavronicolas, and Paul Spirakis. Max-min fair flow control sensitive to priorities. In *Proceedings of the 2nd International Conference on Principles of Distributed Systems*, pages 45–59, 1999.
[13] F. Fazel, M. Fazel, and M. Stojanovic. Random access compressed sensing for energy-efficient underwater sensor networks. *IEEE Journal on Selected Areas in Communications*, 29(8):1660–1670, 2011.



(a) Total preserved priority. (b) Total preserved data items. (c) Total energy cost. (d) Priority-cost ratio.

Fig. 12: Varying number of data packets d_i at DGs under general energy model. Here, $m_i = 50$ and $E_i = 20\%$ of MFE.

[14] R. Ghaffarivardavagh, S. S. Afzal, O. Rodriguez, and F. Adib. Ultra-wideband underwater backscatter via piezoelectric metamaterials. In *Proc. of the ACM SIGCOMM*, 2020.

[15] S. M. Ghoreyshi, A. Shahrabi, and T. Boutaleb. An efficient auv-aided data collection in underwater sensor networks. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 2018.

[16] A V Goldberg and R E Tarjan. A new approach to the maximum flow problem. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, pages 136–146, 1986.

[17] S. Guo, C. Wang, and Y. Yang. Joint mobile data gathering and energy provisioning in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 13(12):2836–2852, 2014.

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

[19] Xiang Hou, Zane Sumpter, Lucas Burson, Xinyu Xue, and Bin Tang. Maximizing data preservation in intermittently connected sensor networks. In *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2012)*. Short Paper.

[20] S. Hsu, Y. Yu, and B. Tang. dre^2: Achieving data resilience in wireless sensor networks: A quadratic programming approach. In *Proc. of IEEE MASS*, 2020.

[21] H. Huang, A. V. Savkin, M. Ding, and C. Huang. Mobile robots in wireless sensor networks: A survey on tasks. *Computer Networks*, 148:1–19, 2019.

[22] H. Huang, A. V. Savkin, M. Ding, and C. Huang. Mobile robots in wireless sensor networks: A survey on tasks. *Computer Networks*, 148:1–19, 2019.

[23] J. Jang and F. Adib. Underwater backscatter networking. In *Proc. of the ACM SIGCOMM*, 2019.

[24] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras. Applications of wireless sensor networks: An up-to-date survey. *Applied System Innovation*, 3(1), 2020.

[25] Hoon Kim. Priority-based qos mac protocol for wireless sensor networks. In *IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2009)*.

[26] Dexter Kozen. Lexicographic flow. Technical report, Computing and Information Science, Cornell University, June 2009.

[27] Raju Kumar, Riccardo Crepaldi, Hosam Rowaihy, Albert F. Harris III, Guohong Cao, Michele Zorzi, and Thomas F. La Porta. Mitigating performance degradation in congested sensor networks. *IEEE Transactions on Mobile Computing*, 7(6):682–697, June 2008.

[28] B. Li, J. Springer, G. Bebis, and M. H. Gunes. Review: A survey of network flow applications. *J. Netw. Comput. Appl.*, 36(2):567–581, mar 2013.

[29] 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.

[30] 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*.

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

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

[33] M. Noormohammadpour and C. S. Raghavendra. Datacenter traffic control: Understanding techniques and tradeoffs. *IEEE Communications Surveys & Tutorials*, 20(2):1492–1525, 2018.

[34] Vilfredo Pareto. Pareto principle. https://en.wikipedia.org/wiki/Pareto_principle.

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

[36] M. Rani, S. B. Dhok, and R. B. Deshmukh. A systematic review of compressive sensing: Concepts, implementations and applications. *IEEE Access*, 6:4875–4894, 2018.

[37] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin. Wireless sensor networks: A survey on recent developments and potential synergies. *J. Supercomput.*, 68(1):1–48, apr 2014.

[38] J. Rivera, Y. Chen, and B. Tang. On the performance of nash equilibria of data preservation in base station-less sensor networks. In *IEEE MASS 2023*.

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

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

[41] B. Tang, H. Ngo, Y. Ma, and B. Alhakami. dao^2: Overcoming overall storage overflow in intermittently connected sensor networks. *IEEE/ACM Transactions on Networking*, 2023.

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

[43] C. Wang, J. Li, Y. Yang, and F. Ye. Combining solar energy harvesting with wireless charging for hybrid wireless sensor networks. *IEEE Transactions on Mobile Computing*, 17(3):560–576, 2018.

[44] L. Wang, Y. Yang, D. K. Noh, H. K. Le, J. Liu, T. D. Abdelzaher, and M. Ward. Adaptsens: An adaptive data collection and storage service for solar-powered sensor networks. In *2009 30th IEEE Real-Time Systems Symposium*, pages 303–312, 2009.

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

[46] A. Wichmann, T. Korkmaz, and A. S. Tosun. Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks. *IEEE Transactions on Mobile Computing*, 17(6):1429–1441, 2018.

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

[48] X. Xue, X. Hou, B. Tang, and R. Bagai. Data preservation in intermittently connected sensor networks with data priority. In *2013 IEEE International Conference on Sensing, Communications and Networking (SECON)*, pages 122–130, 2013.

[49] H. Yedidsion, A. Banik, P. Carmi, M. J. Katz, and M. Segal. Efficient data retrieval in faulty sensor networks using a mobile mule. In *Proc. of WiOpt 2017*.

[50] Y. Yu, S. Hsu, A. Chen, Y. Chen, and B. Tang. Truthful and optimal data preservation in base station-less sensor networks: An integrated game theory and network flow approach. *ACM Trans. Sen. Netw.*, 20(1), oct 2023.