

Priority-Based Data Preservation in Challenging Environments: A Maximum Weighted Flow Approach

Giovanni Rivera¹ and Bin Tang²

Abstract—We address the challenge of preserving large volumes of sensory data with varying priorities in resource-constrained and challenging environments such as underwater exploration and climate monitoring. We establish a new algorithmic framework called *priority-based data preservation (PDP)*, which comprises two network flow problems that have not been adequately studied. We refer to them as *maximum weighted flow with uniform cost (MWF-U)* and *maximum weighted flow with heterogeneous cost (MWF-H)*, respectively. We design integer linear programs, efficient optimal, approximation, heuristic, and distributed algorithms to solve MWF-U and MWF-H. Extensive simulations show that our algorithms outperform classical maximum-flow algorithms (e.g., Edmonds-Karp and push-relabel) that do not consider data priority, preserving between 33.29% and 66.7% more total data priority.

Index Terms—Maximum weighted flow, data preservation, challenging environment, data priority, network flow.

I. INTRODUCTION

IN THIS paper, we focus on emerging sensing applications, including underwater or deep-sea exploration [16], deep space exploration [64], underground pipeline or mine monitoring [47], and volcano eruption monitoring [17]. One common characteristic of these applications is that they are all deployed in harsh, remote, or unattended areas, and high-power data-collecting base stations cannot be deployed in the field. The primary function of such a system, therefore, is to sense the environment and store the sensory data inside the network until uploading opportunities arise to collect the data for further analysis and actuation. These opportunities include periodic visits by unmanned aerial vehicles (UAVs) or drones [37], data mules or robots [30], [55], or autonomous underwater vehicles (AUVs) [24]. Because base stations are not present in the field due to challenging environments, we refer to such sensing systems as *base station-less sensor networks (BSNs)*.

Meanwhile, traditional sensor networks have evolved from single-data-type measurements to multi-functional infrastruc-

tures with heterogeneous data types and modalities. For example, in a recent IoT system for monitoring forest ecology [12], various data types are used to measure the environment, including scalar (e.g., temperature and humidity), vector (e.g., object movement), and multimedia (e.g., cameras and acoustic sensors). In marine monitoring and forecasting [66], multiple underwater communication modalities have been used, including acoustic, optical, and magnetic induction.

Priorities of Sensed Data. Data types are application-specific, and different types have distinct importance levels and contribute differently to the functions of sensing applications. For example, in volcano monitoring [17], [43], the collected data include seismic, infrasonic (low-frequency acoustic), and temperature data. While all data help scientists analyze volcanic activity, seismic and infrasonic data are more crucial than temperature data for interpreting key features of a volcano, such as its scale and magnitude. Another factor contributing to varying data priorities is data quality. In cloud IoT platforms with heterogeneous data sources [49], some data streams are noisier due to the data collection process and thus have lower quality than others. Because processing low-quality data consumes computing resources without significantly improving application performance, it is better to discard it.

Therefore, to balance performance and costs for modern multi-modal, multi-type sensing applications, it is vital to differentiate data by importance level based on their types and quality. In this paper, we refer to such an importance level as *data priorities*. Existing literature refers to this quality as the *value of information* [24], [66] or *data importance* [11].

Data Preservation in the BSN. When events of interest occur in the BSN, sensors close by may collect data more frequently than nodes far away. If drones and robots do not arrive at the BSN field in time to upload data due to unpredictable factors, such as inclement weather, sensor nodes could run out of storage space. We refer to data-generating nodes with depleted storage as *data generators (DGs)*, and to the newly generated data that DGs cannot store locally as *overflow data*. To avoid data loss, overflow data must be offloaded to nearby sensor nodes with available storage (referred to as *storage nodes*) for temporary storage. We call this process *data preservation in BSNs*. The preserved data will then be collected upon the drones' and robots' arrival.

Received 17 February 2025; revised 12 January 2026 and 4 July 2026; accepted 4 September 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor J. Ren. Date of publication 9 September 2026; date of current version 16 September 2026. This work was supported by NSF under Grant CNS-1419952. (Corresponding author: Bin Tang.)

Giovanni Rivera is with the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA 92521 USA (e-mail: grive045@ucr.edu).

Bin Tang is with the Department of Computer Science, California State University Dominguez Hills, Carson, CA 90747 USA (e-mail: btang@csudh.edu).

Digital Object Identifier 10.1109/TON.2026.3732028

2998-4157 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

Authorized licensed use limited to: California State University Dominguez Hills. Downloaded on October 07, 2026 at 17:15:33 UTC from IEEE Xplore. Restrictions apply.

See [37], [48] for different techniques for how drones and robots collect data from the sensor field.

Research Challenges. However, preserving all the overflow data in challenging environments is not always possible. When a BSN has been operating for a long time, the energy and storage capacities of the sensor nodes could reach critical levels.¹ With insufficient storage, some overflow data must be discarded in the data preservation process, causing data loss. With limited battery power, sensor nodes could soon deplete their energy, leading to network partitioning in the BSN and preventing data preservation. When not all overflow data can be successfully preserved in the BSN, finding an energy-efficient data-preservation process that maximizes the total preserved priority, subject to the sensor nodes' storage and energy constraints, becomes a challenging new problem.

Our Contributions. We tackle this challenge by establishing a new algorithmic framework called *priority-based data preservation* (PDP). To solve the PDP, we incorporate BSN characteristics (i.e., sensory data and their priorities, energy and storage of sensor nodes) into a flow network [4], and study a network flow problem called *maximum weighted flow* (MWF). In MWF, multiple source nodes have limited sending capacity, and multiple sink nodes have limited receiving capacity. Unit flows from different source nodes have distinct weights, and flows from any source node can be routed to any sink node. The goal of the MWF is to maximize the total weight of flows from source to sink nodes.

Under a simplified energy model, PDP gives rise to the *MWF with uniform cost* (MWF-U). MWF-U generalizes the classical maximum flow (MF) problem [14], in which each unit of flow has the same weight and the goal is to maximize the total number of units of flow. We design an optimal, efficient algorithm to solve MWF-U. We find that in MWF-U, a maximum weighted flow is a maximum flow, but a maximum flow is not necessarily a maximum weighted flow. Consequently, the well-known *max-flow min-cut theorem* [14], which provides an optimality condition for all the maximum flow algorithms, is not optimal for MWF-U. We propose a *max-weight min-cost theorem* to establish a new optimality condition for MWF-U.

Under a more realistic energy model, PDP yields the *MWF with heterogeneous cost* (MWF-H), where unit flows from different source nodes not only have distinct weights but also cost different amounts of edge capacity. We prove that MWF-H is NP-hard and design a 2-approximation algorithm under some conditions. In MWF-H, the capacity of an edge must be greater than or equal to the *linear combination* (i.e., *weighted sum*) of the unit flows on that edge. As such, a maximum weighted flow need not be a maximum flow to achieve the maximum total weight.

The limited capacities of the source and sink nodes in MWF-U and MWF-H are uniquely derived from PDP: DGs have finite overflow data packets, storage nodes have limited storage capacity, and all are energy-constrained. This raises another interesting question: whether it is possible to send all

available flows from the sources to the sink nodes (if so, MWF becomes trivial). In contrast, the existing MF and MWF [45] problems do not entail such feasibility checks.

We further design a distributed data-preservation algorithm that achieves optimal data priorities with time and message complexities of $O(kn^2)$ and $O(n^2m)$, respectively, where n , m , and k are the numbers of nodes, edges, and DGs, respectively. In extensive simulations, our priority-based data preservation algorithms outperform classical maximum-flow algorithms that do not consider data priority (e.g., Edmonds-Karp [14] and push-relabel [25]) by yielding between 33.29% and 66.7% more total preserved priority.

Paper Organization. Section II discusses the state-of-the-art. In Section III, we formulate the PDP, MWF-U, and MWF-H, and study their feasibility conditions. Section IV and V present efficient optimal and approximation algorithms to solve MWF-U and MWF-H, respectively. Section VI solves PDP by connecting it with MWF-U and MWF-H. Section VII presents the distributed data preservation algorithm. In Section VIII, we compare our algorithms with existing works and discuss the results. Section IX concludes with future research.

II. RELATED WORK

Multi-Commodity Flow (MCF). MCF [4], [46] studies the pairwise routing of distinct types of flows (i.e., commodities) from their respective sources to destinations, and is a *one-to-one* model. MWF is instead an *any-to-any* model, in which flows from *any* source can be routed to *any* sink. As such, the solution techniques for MWF and MCF are very different. The closest MCF problem to ours is maximum weighted MCF (MW-MCF) [23], [33], wherein source-sink pairs have different weights, and the goal is to maximize the total weight of the flows. MW-MCF is APX-hard, which implies that no constant-factor approximation algorithm exists unless $P=NP$. We instead show that MWF-U can be solved optimally and efficiently on general graphs.

Chekuri [10] studied *multicommodity demand flow*, where each source-destination pair has a request demand and a profit, and the goal is to find a set of pairs that maximizes profit while the demands of different pairs are constrained by the edge capacity. The request demands resemble the unit flow costs in MWF-H. However, they assume unsplitable flows, in which each pair's request is either satisfied or unsatisfied. The MWF-H instead assumes splittable flows, allowing some overflow data packets from each source node to be preserved. They focused on a tree topology and proposed a 4-approximation algorithm using path-coloring techniques. Instead, we propose a polynomial-time 2-approximation algorithm for solving the MWF-H on general graphs under certain conditions.

Rothfarb et al. [45] and Exercise 17.15 in [4] are the first to study the maximum weighted flow. MWF-U and MWF-H differ from them due to the limited capacities on the source and sink nodes. As such, it first needs to check whether it is feasible to push all available flows from the sources to the sink nodes in MWF-U and MWF-H.

Data Preservation in the BSN. Various data preservation techniques have been proposed considering that nodes have

¹We 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 [38] or using solar power [54].

TABLE I
SUMMARY OF PRIORITY-BASED FLOW PROBLEMS. MWF: MAXIMUM WEIGHTED FLOW. MPF: MAXIMUM PRIORITY FLOW

	MWF-U and MWF-H	Lexicographic Flow	MPF	Priority-based Max-Min Flow
Setup	Priorities on flows	Priorities on edges	Node-specific edge priorities	Priorities on flows
Goal	Maximize total weight of flows	Maximum priorities on edges	Maximize the value of priority flow	Fairness with priority awareness

limited battery power [28], [29], [51]. When there is insufficient available storage in the BSN to store all the overflow data, an aggregation technique has been proposed, and the problem is modeled as a *traveling salesman placement problem* [50], [52]. Considering that sensor nodes are intelligent and selfish, Rivera et al. [44] studied the performance of Nash Equilibria for data preservation, and Yu et al. [63] motivated selfish sensor nodes to achieve truthful and optimal data preservation using an integrated game-theoretic and network-flow approach.

None of the above works considered data priorities in the data preservation process, which is the topic of this work. Our conference paper [62] studied MWF-U. This paper extended it by considering a more general and realistic energy model, with deeper theoretical and experimental analyses.

Drone-based IoT and Sensor Networks. Autonomous drones have been utilized to monitor and gather data from IoT and sensor networks [8], [15], [37], [48], [56]. For example, Li et al. [37] focused on buffer overflows at ground sensors and studied how to schedule drones to minimize the overall data packet loss. Our work is orthogonal to theirs: to prevent data loss during a buffer overflow, overflow packets are offloaded to nodes with available buffer space. Besides, none of the above works consider preserving data with different priorities when sensor nodes have minimal battery power.

Another line of work studied how to deploy and schedule drones for disaster monitoring to maximize monitoring rewards subject to the drones' battery capacity [57], [58], [59], [60], [61], [65]. They formulated and solved various *team orienteering problems* [9], [53]. Our approach complements theirs by focusing on the sensor nodes' limited battery capacity. Because drones can be conveniently recharged (upon returning to the base station), whereas sensor nodes cannot, our approach is more relevant to data preservation in BSNs.

Priority-Based Techniques. In the theory community, researchers have considered edge priorities in maximum flow problems. Kozen [34] studied 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 [34], a lexicographically maximum flow is not necessarily a maximum flow. Another problem, maximum priority flow, is studied in [2] and [5]. In this problem, each node specifies a priority ordering of the edges leaving it, and flows leaving the node always go through higher-priority edges before lower-priority ones. Neither work considers assigning priority to each flow. The most closely related work is by Fatourou et al. [21]. They studied priority-based max-min fairness, in which each session has a priority, and the goal is to assign each session the maximum possible rate corresponding to its priority. However, they focus on classical max-min fairness theory, which differs from our goal

of maximizing the total weight of all flows. Table I summarizes various priority-based techniques.

A few works in the sensor network community study priority-based data dissemination. Gjanci [24] proposed a distributed heuristic for path-finding in an AUV collecting data with decaying values in underwater sensor networks. However, they did not consider the sensor nodes' energy and storage constraints. Kumar [35] addressed how to deliver data with different importance (priority) in congested sensor networks. Kim [32] proposed a quality-of-service MAC protocol based on data-transmission priority levels. However, it assumes 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.

Security, Data Integrity, and Fault-Tolerance in Harsh Environments. As BSNs are deployed in challenging and sometimes adversarial environments (e.g., battlefield), preventing data manipulation and false data injection from malicious nodes becomes critical. By exploiting the unique characteristics of the BSN, our system can prevent false priority injection and data manipulation using three existing approaches. First, legitimate sensor readings usually exhibit spatial and temporal correlations, which can be used to perform correlation analysis to prevent fault-data-injection attacks [36]. Second, because BSNs use multi-hop routing, intermediate nodes along the data-offloading path can perform basic validity checks on the data packets they forward [40]. Third, because wireless communication is a broadcast medium, a sensor node can "overhear" messages from all of its neighboring nodes. This can be used to develop a reputation system that detects malicious nodes among others [22]. Our system assumes that such security mechanisms are in place to ensure flow optimization and that preservation results are not distorted. Finally, existing research has studied how to detect faulty nodes based on a mismatch between sensing readings and distance readings [26] and how to detect communication link failures using a Kalman-consensus filtering approach [39]. Their research complements ours: once faulty nodes or links are detected, our algorithms can be activated to recompute data preservation.

In summary, existing weighted-flow and network utility maximization studies aim to maximize weighted throughput under capacity constraints, whereas our work focuses on preserving sensor data with heterogeneous priorities under joint energy and storage constraints. Unlike minimum-cost flow formulations, where costs are associated with edges and the objective is to minimize transportation cost, the proposed MWF formulations assign priorities to source-generated data and maximize the total preserved importance. Priority-aware routing and packet scheduling approaches similarly consider differentiated traffic classes but typically rely on heuristic prioritization policies and offer no optimality guarantees. In

contrast, our framework establishes a flow-theoretic foundation for priority-aware data preservation, including polynomial-time optimality results for MWF-U, a Max-Weight Min-Cost equivalence theorem, and approximation guarantees for the NP-hard MWF-H.

III. PRELIMINARY

We formally define the PDP, MWF-U, and MWF-H. We also study the feasibility conditions of MWF-U and MWF-H.

A. Priority-Based Data Preservation (PDP)

Network Model. The BSN is represented as an undirected graph $G(V, E)$, where $V = \{1, 2, \dots, |V|\}$ is n uniformly deployed sensor nodes, and E is the set of m edges. An edge connects two sensor nodes if they are within each other's transmission range and can thus communicate directly. There are k data generators (DGs), denoted as $V_d = \{DG_1, DG_2, \dots, DG_k\} \subset V$. The remaining nodes are storage nodes, denoted V_s . The sensory data are modeled as a sequence of raw data items, each of unit size. Data items generated at DG_i all have priority value v_i , indicating their importance or quality in a specific application. Let d_i denote the amount of overflow data at DG_i that needs to be preserved. 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 $m_i = 0$; if $i \in V_s$, then $m_i > 0$. Let $d = \sum_{i=1}^k d_i$ and $m = \sum_{i \in V_s} m_i$. Our techniques apply to both cases of $m \geq d$ and $m < d$.

In this work, we assume that data priorities are known a priori and remain fixed during the preservation interval. This assumption is reasonable for many environmental sensing applications, where domain experts determine the relative importance of different data modalities (e.g., seismic versus temperature measurements), and priorities change slowly relative to the data-preservation timescale.

Energy Model. Each sensor node i has finite and unreplenishable initial energy E_i . We consider two energy models.

- *Uniform Energy Model.* Any node involved in preserving a data item, whether it's sending, relaying, or finally receiving, costs the same amount and one unit of energy. The energy consumption of preserving a data item thus equals the number of nodes involved in the process. This is a good approximation for uniformly deployed sensor nodes.

- *General Energy Model.* The first-order radio model [27] is a more general energy model for wireless communication. For node i sends a b -bit data to node j over their distance $l_{i,j}$, the *transmission energy* incurred at i is $E_t(b, l_{i,j}) = E_{elec} \times b + \epsilon_{amp} \times b \times l_{i,j}^2$, and the *receiving energy* incurred at j is $E_r(k) = E_{elec} \times k$. Here, $E_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $\epsilon_{amp} = 100 pJ/bit/m^2$ is the energy consumption per bit on the transmit amplifier [52].

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

- a subset of data items $\mathcal{D} \subseteq D$ to preserve, and
- *preservation function* $r : \mathcal{D} \rightarrow V_s$, indicating that $D_j \in \mathcal{D}$ is offloaded from $s(j)$ to its *destination node* $r(j) \in V_s$, and

TABLE II
NOTATION SUMMARY

Notation	Explanation
$G(V, E)$	BSN graph with $ V = n$ sensor nodes and $ E = m$ edges
E_i	Initial energy of node i
m_i	Storage capacity of node i
m	Total storage spaces available, $m = \sum_{i \in V_s} m_i$
E'_i	Remaining energy of node i after data preservation
V_d	Set of $k = V_d $ DGs
DG_i	i^{th} DG, $1 \leq i \leq k$
V_s	Set of storage nodes
d_i	Number of overflow data items at DG_i
v_i	Priority of each data item at DG_i
d	Total number of overflow data items, $d = \sum_{i=1}^k d_i$
D	The entire set of d overflow data items
D_j	j^{th} overflow data item
$\mathcal{D}$	$\mathcal{D} \subseteq D$ is the set of preserved data items
x_{ij}	Energy cost of node i in preserving D_j
$s(j)$	DG node of data item D_j
$r(j)$	Destination node of D_j ; it is a storage node
$\mathcal{P}_j$	Preservation path of D_j
S	$s_i \in S$ is the set of source nodes in a flow network
T	$t_j \in T$ is the set of sink nodes in a flow network
c_i	Cost of s_i 's unit flow in MWF-H, $c_i = 1$ in MWF-U
$ca(u, v)$	Capacity of the edge (u, v)
$f(u, v)$	Number of unit flows on edge (u, v)
$f_i(u, v)$	Number of unit flows of s_i on edge (u, v)
$\mathcal{V}_f$	The total weight of the flow f
$\mathcal{V}_f^m$	The maximum value of $\mathcal{V}_f$; $\mathcal{V}_f^m = \sum_{i=1}^k (d_i \cdot v_i)$

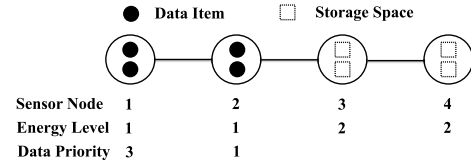


Fig. 1. Illustrating the PDP problem.

- *preservation path* of $D_j \in \mathcal{D}$, referred to as $\mathcal{P}_j : s(j), \dots, r(j)$, 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 offloading D_j from $s(j)$ to $r(j)$. Let E'_i denote i 's energy level after the data preservation; $E'_i = E_i - \sum_{D_j \in \mathcal{D}} x_{ij}$. The objective of PDP is to find a subset of data items $\mathcal{D} \subseteq D$, the corresponding storage node $r(j)$ of $D_j \in \mathcal{D}$, and the preservation path $\mathcal{P}_j$ of D_j between $s(j)$ and $r(j)$, 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 node's energy constraint, i.e., $E'_i \geq 0, \forall i \in V$, and storage capacity constraint, i.e., $|\{j \mid r(j) = i, D_j \in \mathcal{D}\}| \leq m_i, \forall i \in V_s$. Table II 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; each has 2 available storage spaces and an initial energy level of 2. The priorities of the data items in nodes 1 and 2 are 3 and 1, respectively. Under the uniform energy model, the optimal solution, which is also a maximum flow, is to offload one data item from node 1 to node 3, preserving the priority of 3.

For the general energy model, we consider a simplified version where sending or receiving a data item costs 0.5 units

of a sensor node's energy. That is, the energy consumption of preserving a data item equals the number of edges it traverses. The optimal solution is offloading one data item from node 1 to node 3, resulting in a total preserved priority of 3. The maximum flow would be to offload 2 data items from node 2 to node 3, yielding a total preserved priority of 2. $\square$

Time-Critical Scenarios. Our model can be extended to accommodate time-critical, deadline-constrained data by factoring data freshness into a time-slotted data preservation process. We can assume that data with hard deadlines has the highest priority, and define the priority value of data packets at time slot t as $v_i(t) = v_i/(t - u(t))$, where $u(t)$ is the time the packets are generated at DG_i . For packets with a delivery deadline t_d , they will be discarded if $t - u(t) > t_d$.

B. MWF-U and MWF-H

Let $G = (V, E)$ be a flow network, a directed graph with k source nodes $S = \{s_1, s_2, \dots, s_k\} \subset V$ and l sink nodes $T = \{t_1, t_2, \dots, t_l\} \subset V$. The rest of the nodes in $V \setminus S \cup T$ are called *transshipment nodes*. Each unit flow represents a data item to be preserved in PDP and cannot be further split; a unit flow starting from a source node can end at any sink node. The *capacity of an edge* $(u, v) \in E$, denoted as $ca(u, v)$, represents the maximum amount of unit flows that can pass through (u, v) . Each unit flow from s_i has a *weight* of v_i , indicating the importance of the flows generated by s_i . Let $d = \sum_{i=1}^k d_i$ and $m = \sum_{j=1}^l m_j$. We use a function $f_i : E \rightarrow \mathbb{N}$ (nonnegative integers) to represent the amount of unit flows of source node s_i on $(u, v) \in E$.

Definition 1 (Sending (Receiving) Capacity of Source (Sink) Nodes): The *sending capacity* d_i of source node $s_i \in S$ is the maximum number of unit flows it can send. The *receiving capacity* m_j of sink node $t_j \in T$ is the maximum number of unit flows it can receive. In PDP, d_i represents the number of overflow data items of DG_i and m_j the storage capacity of storage node t_j . $\square$

MWF-U. In MWF-U, each unit flow uses one amount of edge capacity. Given a valid flow of all the k source nodes $f = \{f_1, \dots, f_k\}$ and any edge (u, v) , it must satisfy the following four constraints in MWF-U:

- 1). *Source node-specific flow conservation:* $\sum_{u \in V} f_i(u, v) = \sum_{v \in V} f_i(v, u)$, $\forall v \in V \setminus (S \cup T)$, $1 \leq i \leq k$.
- 2). *Uniform edge capacity constraint:* $f(u, v) = \sum_{i=1}^k f_i(u, v) \leq ca(u, v)$, $\forall (u, v) \in E$. This follows the uniform energy model in PDP, where sending, relaying, and receiving cost the same amount of energy.
- 3). *Source node capacity constraint:* For any source node s_i , its *net flow* $|f_i| \leq d_i$ is the number of unit flows it sends out; $|f_i| = \sum_{u \in V} (f_i(s_i, u) - f_i(u, s_i))$. In PDP, $|f_i|$ is the number of data items from DG_i that are preserved.
- 4). *Sink node capacity constraint:* For any sink node t_j , its *net flow* $|f_j|$ is the number of unit flows it receives: $\sum_{u \in V} (f_j(u, t_j) - f_j(t_j, u))$; $0 \leq |f_j| \leq m_j$. In PDP, $|f_j|$ is the number of data items preserved at t_j .

Definition 2 (Total Weight of a Flow f): The *total weight* of a flow f , denoted as $\mathcal{V}_f$, is the sum of weights of the net flows out of all the source nodes. That is, $\mathcal{V}_f = \sum_{i=1}^k (v_i \times |f_i|) = \sum_{i=1}^k v_i \times \sum_{u \in V} (f(s_i, u) - f(u, s_i))$. $\mathcal{V}_f$

has its maximum value of $\mathcal{V}_f^m = \sum_{i=1}^k (d_i \cdot v_i)$, when all the flows from all the source nodes are successfully sent. $\square$

The goal of the MWF-U is to find a valid flow f from S to T such that $\mathcal{V}_f$ is maximized. We present an optimal and efficient algorithm to solve MWF-U in Section IV.

Example 2: Fig. 2(a) shows a flow network with two source nodes s_1 and s_2 and two sink nodes t_1 and t_2 , with $v_1 = 3$, $v_2 = 1$, $d_1 = d_2 = 2$, and $m_1 = m_2 = 2$. All the edge capacities are 2. The optimal MWF-U solution is for s_1 to send two unit flows to t_1 , with $\mathcal{V}_f = 6$. $\square$

MWF-H. MWF-H considers that different unit flows may use different amounts of edge capacity. This is based upon the general energy model in the PDP, in which a sensor node spends different amounts of transmission energy when sending data items to nodes at different distances. Define the *cost of a unit flow* as the amount of edge capacity this unit flow uses; a unit flow from s_i uses $c_i \in \mathbb{N}^+$ (positive integer) amounts of edge capacity. When $c_i = 1$, it becomes the uniform edge capacity constraint in MWF-U. A valid flow $f = \{f_1, \dots, f_k\}$ in MWF-H satisfies all the constraints of MWF-U except for 2), which is replaced with

$$f(u, v) = \sum_{i=1}^k (f_i(u, v) \cdot c_i) \leq ca(u, v). \quad (1)$$

We refer to it as *generalized edge capacity constraint* (GECC), which shows that the capacity of an edge must be greater than or equal to the weighted sum of the unit flows on that edge. The goal of the MWF-H is to find a valid flow f from S to T to maximize $\mathcal{V}_f$ under these four constraints.

Example 3: For Fig. 2(a), consider an MWF-H instance by adding $c_1 = 2$ and $c_2 = 1$. Its optimal solution is for s_1 to send one unit of flow to t_1 , giving $\mathcal{V}_f = 3$. $\square$

C. Feasibility Conditions for MWF-U and MWF-H

Before solving MWF-U and MWF-H for any flow network $G(V, E)$, we need to determine whether they are feasible; that is, not all unit flows at source nodes can be sent. Thus, we need to find the maximum flow in G for MWF-U and MWF-H.

Capacity-Restricted Flow Network (CFN) G' . To find the maximum flow in G , we need to represent the source node's sending and the sink node's receiving capacities as edge capacities in the flow network. We thus transform G into another *capacity-restricted flow network* (CFN) $G'(V', E')$ as follows. We add into G a super source s and directed edges (s, s_i) with capacity $ca(s, s_i) = d_i$, $1 \leq i \leq k$, and add a super sink t and directed edges (t_j, t) with capacity $ca(t_j, t) = m_j$, $1 \leq j \leq l$. That is, $V' = \{s\} \cup V \cup \{t\}$, $E' = \bigcup_{i=1}^k (s, s_i) \cup E \cup \bigcup_{j=1}^l (t_j, t)$. Fig. 2(b) shows the G' transformed from the G in Fig. 2(a). CNFs are also used in Section IV and V to compute the maximum weighted flow in MWF-U and MWF-H once feasibility is established.

Feasible Augmenting Path in G' . Given a CFN G' with a current flow of f , any augmenting path p from s to t in its residual graph G'_f must go through exactly one source node s_i and one sink node t_j . However, due to the limited sending (receiving) capacities of the source (sink) nodes, not all augmenting paths can be used to further increase flows.

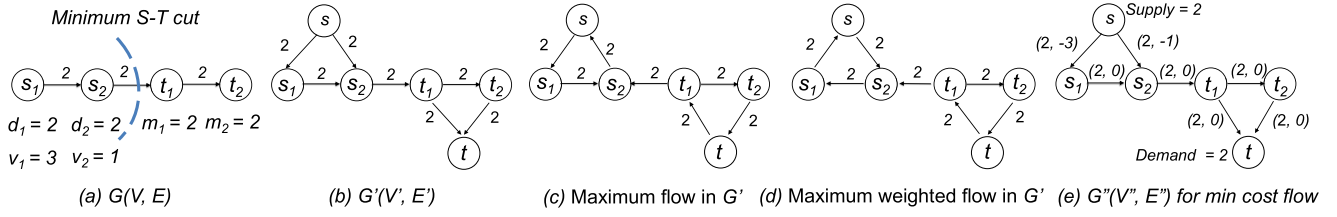


Fig. 2. (a) flow network G , (b) capacity-restricted flow network G' , (c) residual graph after running Algo. 1 on G' , (d) residual graph after running Algo. 2 on G' , and (e) flow network G'' for minimum cost flow (in each parenthesis, the first value is edge capacity and the second is edge cost).

Algorithm 1 Maximum Flow Algorithm for MWF-U

Input: a CFN $G'(V', E')$ with S, T, d_i, m_j .

Output: the maximum flow $|f|$.

Notations: f : current flow from s to t in G' ;

$|f|$: the number of unit flows in f ;

G'_f : residual graph of G' with flow f ;

1: $\forall (u, v) \in E', f(u, v) = 0, |f| = 0, G'_f = G'$;

2: **while** (G'_f has a shortest FAP p) **do**

3: Find the source s_i and the sink t_j p goes through;

4: Let δ_p be the minimum residual capacity along p ;

5: $\Delta_p = \min(d_i, m_j, \delta_p)$

6: **for** (each edge $(u, v) \in p$) **do**

7: $f(u, v) += \Delta_p, f(v, u) -= \Delta_p$; // Augment flow f

8: **end for**

9: $|f| += \Delta_p, d_i -= \Delta_p, m_j -= \Delta_p$;

10: **end while**

11: **return:** $|f|$.

Definition 3 (Feasible Augmenting Path (FAP)): Given a CFN G' with a current flow of f , a *feasible augmenting path* p in its residual graph G'_f is an augmenting path that can be used to further increase the flow from s to t . Assume p goes through a source node s_i and a sink node t_j ; it must be that s_i still has unit flows to send, and m_j still has capacity to receive some unit flows. That is, $f(s, s_i) < d_i$ and $f(t_j, t) < m_j$. ■

MWF-U. Given a CFN G' for an MWF-U instance, Algo. 1 computes its maximum flow by iteratively finding the shortest FAP. In each round, it first finds the shortest FAP p , and the source s_i and the sink t_j that p passes through (lines 2 and 3). Next, it computes the amount of unit flows Δ_p that can be increased along p (lines 4 and 5). It then augments the flow and updates the maximum flow amount $|f|$, d_i , and m_j (lines 6-9). This continues until no further FAPs are found, and returns $|f|$. If $|f| < d$, the MWF-U is feasible. The time complexity of Algo. 1 is $O(|V'| \cdot |E'|^2)$. It is based on the classical Edmonds-Karp maximum flow algorithm [14], except that it must check that a shortest augmenting path is an FAP for MWF-U. We refer to Algo. 1 as *Modified Edmonds-Karp*.

Example 4: Fig. 2(c) shows the residual graph after executing Algo. 1 on G' in Fig. 2(b). It pushes two unit flows along a FAP: s, s_2, t_1 , and t , yielding a maximum flow of 2. As it is less than 4, it is feasible. □

Theorem 1: Algo. 1 computes maximum flow for MWF-U.

Proof: For Algo. 1, its maximum flow for MWF-U is based on the well-known max-flow min-cut theorem [14]. The theorem states that the maximum amount of flow passing from

a source node s to a sink node t is equal to the minimum capacity of all possible cuts that separate s from t . For Algo. 1, this happens when no feasible augmenting path can be found from s to t . ■

MWF-H. We can formulate the maximum flow problem in MWF-H as an integer linear program as follows.

$$\max \sum_{i=1}^k \sum_{u \in V} (f_i(s_i, u) - f_i(u, s_i)) \quad (2)$$

$$\text{s.t.} \sum_{u \in V} (f_i(s_i, u) - f_i(u, s_i)) \leq d_i, 1 \leq i \leq k \quad (3)$$

$$\sum_{u \in V} (f_j(u, t_j) - f_j(t_j, u)) \leq m_j, 1 \leq j \leq l \quad (4)$$

$$\sum_{u \in V} f_i(u, v) = \sum_{u \in V} f_i(v, u), v \in V \setminus (S \cup T) \quad (5)$$

$$\sum_{i=1}^k (f_i(u, v) \cdot c_i) \leq ca(u, v), (u, v) \in E \quad (6)$$

Objective (2) maximizes the total number of flows. Ineqs. (3) and (4) indicate the source and sink node capacity constraints, respectively. Eqns. (5) and (6) enforce source-node-specific flow conservation and the GECC, respectively.

Example 5: The maximum flow in Example 3 is that s_2 sends two unit flows to t_1 , giving $\mathcal{V}_f = 2$. □

For the same MWF-H instance, Example 3 achieves maximum weighted flow but not maximum flow, while Example 5 achieves maximum flow but not maximum weighted flow. We thus have the following observation.

Observation 1: In MWF-H, the maximum weighted flow is not necessarily a maximum flow, and a maximum flow is not necessarily a maximum weighted flow.

IV. OPTIMAL ALGORITHM FOR MWF-U

In this section, we first solve MWF-U optimally and efficiently, and then provide an optimality condition for it.

A. Greedy Optimal Algorithm (GOA)

GOA viz. Algo. 2 runs on the CFN G' . It iteratively pushes as much unit flow as possible from s to t that passes through each source node, sorted in non-ascending order of their weights. This continues until no further feasible augmenting paths (FAP) can be found, at which point it returns the flow f and its total weight $\mathcal{V}_f$. Its running time is $O(k \cdot |V| \cdot |E|^2)$.

Example 6: Fig. 2(d) shows the residual graph after applying Algo. 2 on Fig. 2(b). It pushes two unit flows along the feasible

Algorithm 2 Greedy Optimal Algorithm (GOA) for MWF-U**Input:** a CFN $G'(V', E')$, S , T , d_i , v_i , and m_j .**Output:** flow f and its total weight $\mathcal{V}_f$.**Notations:** G' : a capacity-restricted flow network; f : current flow from s to t in G' ; G'_f : residual graph of G' with flow f ;

1: Sort source nodes in non-ascending order of their weights.

WLOG, $v_1 \geq v_2 \geq \dots \geq v_k$;2: $\forall (u, v) \in E'$, $\mathcal{V}_f = f(u, v) = 0$, $G'_f = G'$;3: **for** ($1 \leq i \leq k$) **do**4: **while** (G'_f has a shortest FAP p going through s_i) **do**5: Find the sink node t_j that p goes through;6: Let δ_p be the minimum residual capacity along p ;7: $\Delta_p = \min(d_i, m_j, \delta_p)$ 8: **for** (each edge $(u, v) \in p$) **do**9: $f(u, v) += \Delta_p$, $f(v, u) -= \Delta_p$;10: **end for**11: $\mathcal{V}_f += \Delta_p \cdot v_i$, $d_i -= \Delta_p$, $m_j -= \Delta_p$;12: **end while**13: **end for**14: **return:** f and $\mathcal{V}_f$.

augmenting path: s, s_1, s_2, t_1 , and t , meaning s_1 sends two unit flows of $\mathcal{V}_f = 6$ to t_1 in Fig. 2(a). The Modified Edmonds-Karp instead gives $\mathcal{V}_f = 2$ (Fig. 2(c)). $\square$

Lemma 1: GOA yields a maximum flow.

Proof: The GOA is the Modified Edmonds-Karp (i.e., Algo. 1) executed in a non-ascending order of source nodes' weights. As each unit flow reduces one edge capacity in the residual graph, when no more FAP can be found, it yields a maximum flow following the max-flow min-cut theorem [14]. $\blacksquare$

B.: Optimality of GOA

To prove that GOA is optimal, we use a constructive approach to show that any optimal MWF-U solution cannot yield flows with total weight exceeding that of GOA.

Notations. Let $P = \{P_1, P_2, \dots, P_k\}$ be the flow solution yielded by GOA (Algo. 2), indicating that s_i has P_i amount of net flow in GOA. Let $O = \{O_1, O_2, \dots, O_k\}$ be the optimal solution, indicating that s_i has O_i amount of net flow in the optimal solution. WLOG, we assume that $v_1 \geq v_2 \geq \dots \geq v_k$. Let $\Lambda = \{s_{\lambda(1)}, s_{\lambda(2)}, \dots, s_{\lambda(a)}\}$, with $1 \leq \lambda(1) < \lambda(2) < \dots < \lambda(a) \leq k$, 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 $\Xi = \{s_{\xi(1)}, s_{\xi(2)}, \dots, s_{\xi(b)}\}$, with $1 \leq \xi(1) < \xi(2) < \dots < \xi(b) \leq k$, 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 $a + b \leq k$.

Constructive Algorithm. According to Lemma 1, the total surplus of Λ equals the total deficit of Ξ , that is, $\sum_{i=1}^a (P_{\lambda(i)} - O_{\lambda(i)}) = \sum_{j=1}^b (O_{\xi(j)} - P_{\xi(j)})$. We thus design Algo. 3 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 4 (Matching Set of $s_{\lambda(i)}$): The matching set of $s_{\lambda(i)} \in \Lambda$, denoted as $\{s_{\xi(l_i)}, s_{\xi(l_i+1)}, \dots, s_{\xi(h_i)}\} \subseteq \Xi$, where

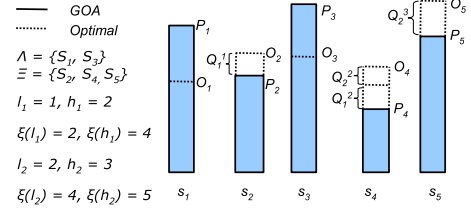


Fig. 3. An example with five source nodes: $s_1, \dots, s_5$ illustrates how Algo. 3 works. It maps $\Xi = \{s_2, s_4, s_5\}$ to $\Lambda = \{s_1, s_3\}$. The matching set of s_1 is $\{s_2, s_4\}$, with $P_1 - O_1 = Q_1^1 + Q_2^1$. The matching set of s_3 is $\{s_4, s_5\}$, with $P_3 - O_3 = Q_2^2 + Q_3^2$. Note that $v_1 \geq v_2 \geq \dots \geq v_5$.

$1 \leq l_i \leq h_i \leq b$, is a subset of Ξ with adjacent elements whose total matching amount is $(P_{\lambda(i)} - O_{\lambda(i)})$. 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 $\sum_{j=l_i}^{h_i} Q_i^j = (P_{\lambda(i)} - O_{\lambda(i)})$. $\square$

Algo. 3 computes $s_{\lambda(i)}$'s matching set and the matching amount of each element in this matching set. It achieves this by rearranging the total deficit of Ξ and mapping Ξ to Λ . In particular, it finds $s_{\lambda(i)}$'s matching set and calculating Q_i^j , the corresponding matching amount from $s_{\xi(j)}$ to $s_{\lambda(i)}$, such that this matching set's total matching amount equals $s_{\lambda(i)}$'s surplus. 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 as 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. 3 is an example with five source nodes that illustrates the execution of Algo. 3. This algorithm 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 $1 \leq i \leq j - 1$, Algo. 2 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 the 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 the fact that GOA is a greedy algorithm that finds the maximum amount of data to distribute from $s_{\xi(l_j)}$ at this stage. $\blacksquare$

Theorem 2: GOA is an optimal algorithm for the MWF. That is, it finds a flow with the maximum total weight.

Proof: By way of contradiction, assume that GOA is not optimal; i.e., $\sum_{i=1}^k O_i \times v_i > \sum_{i=1}^k P_i \times v_i$. Then,

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

Algorithm 3 A Constructive Algorithm Mapping Ξ to Λ

Input: Optimal flows O_i and GOA flows P_i , $1 \leq i \leq k$.
Output: l_i , h_i , $\{Q_i^l, Q_i^{l+1}, \dots, Q_i^{h_i}\}$, $1 \leq i \leq a$.
Notations: $alloc$: total matching amount 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 ( $1 \leq i \leq a$ ) do
4:    $j = l_i$ ;  $flag = true$ ;
5:   while ( $alloc < (P_{\lambda(i)} - O_{\lambda(i)})$ ) do
6:      $j++$ ;
7:      $Q_i^j = O_{\xi(j)} - P_{\xi(j)}$ ;  $alloc = alloc + Q_i^j$ ;
8:      $flag = false$ ;
9:   end while
10:   $h_i = j$ ;
11:  if ( $flag == true$ ) then
12:     $Q_i^j = P_{\lambda(i)} - O_{\lambda(i)}$ ;
13:  else
14:     $Q_i^j = (O_{\xi(j)} - P_{\xi(j)}) - (alloc - (P_{\lambda(i)} - O_{\lambda(i)}))$ ;
15:  end if
16:  if ( $alloc == (P_{\lambda(i)} - O_{\lambda(i)})$ ) then
17:     $l_{i+1} = j + 1$ ;
18:     $alloc = O_{\xi(l_{i+1})} - P_{\xi(l_{i+1})}$ ;
19:  else
20:     $l_{i+1} = j$ ;
21:     $alloc = alloc - (P_{\lambda(i)} - O_{\lambda(i)})$ ;
22:  end if
23: end for
24: return:  $[l_1, l_2, \dots, l_a]$ ,  $[h_1, h_2, \dots, h_a]$ , and  $Q_i^j$ .
```

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). ■

Discussions. Although GOA is an augmenting-path maximum flow algorithm executed in the non-ascending order of the weights of the source nodes, its optimality proof is non-trivial, as greedy algorithms do not yield optimal solutions for many combinatorial problems. The greedy method yields optimal solutions when the problem has a combinatorial structure known as “matroids” [14]. They generalize linear independence in vector spaces to problems for which the greedy algorithm yields an optimal solution. Algo. 3 demonstrates in a constructive way that MWF-U has a matroid-like structure, thus differing significantly from other greedy algorithms.

C. Optimality Condition for MWF-U

Max-Flow Min-Cut Theorem. The well-known *max-flow min-cut theorem* [14] gives the optimality condition for the maximum flow algorithms. It states that a flow is maximum if and only if the residual graph contains no augmenting path, which yields a minimum-capacity cut. However, this theorem is no longer sufficient for the optimality of an

MWF-U algorithm. We first define the cut in the multi-source, multi-sink MWF scenario.

Definition 5 (*S-T Cut in MWF*): Give a flow network $G(V, E)$ with source nodes S and sink nodes T , an *S-T cut* partitions V into S' and T' s.t. $S \subseteq S'$ and $T \subseteq T'$. The capacity of the cut is $ca(S, T) = \sum_{u \in S'} \sum_{v \in T'} ca(u, v)$. □

For example, Fig. 2(a) shows the minimum *S-T* cut of G , with $ca(S, T) = ca(s_2, t_1) = 2$. Thus, from the max-flow min-cut theorem, the maximum flow allowed in G is 2. However, the theorem cannot distinguish between maximum flow and maximum weighted flow. Although Fig. 2(c) and (d) are both maximum flows, only Fig. 2(d) is a maximum weighted flow. Therefore, we next establish a valid optimality condition for any MWF-U optimal algorithm, including GOA.

Max-Weight Min-Cost Theorem. Minimum cost flow (MinCF) [4] generalizes the maximum flow by considering that each edge has both a capacity and a cost, and its goal is to send a fixed amount of flow (i.e., the supply at s and the demand at t) from s to t at minimum cost. It can be solved efficiently and optimally using cycle-canceling and successive shortest-path algorithms [4].

To run MinCF, we convert the CFN $G'(V', E')$ to another flow network $G''(V'' = V', E'' = E')$ as follows. We first compute the maximum flow in G' (using Modified Edmonds-Karp) and denote it as f^m . We then set both the supply at s and the demand at t as $|f^m|$. Next, while keeping the capacities of all the edges E' unchanged, we add the cost of $(s, s_i) \in E'$ as $-v_i$, the negative of s_i 's weight, and the cost of all other edges in E' as 0. With this transformation, the G' in Fig. 2(b) is changed to G'' in Fig. 2(e). We give the following theorem.

Theorem 3: Max-Weight Min-cost. A flow f is a MWF in G' if and only if f is a MinCost in G'' .

Proof: If $f = \{f_1, \dots, f_k\}$ in G' is a maximum weighted flow, for any other maximum flow f' in G' , there exists a $1 \leq j \leq k$ such that $f_j > f'_j$ while $f_i = f'_i$ for $1 \leq i < j$. This is because a) $|f| = |f'| = |f^m|$ (Lemma 1), b) $v_1 \geq v_2, \dots, \geq v_k$, and c) f always pushes flows into source nodes s_i in the non-ascending order of their weights. In G'' , as the cost of an edge (s, v_i) is $-v_i$, this results in a flow with a cost not higher than that of any other maximum flow. Therefore, f is a MinCF in G'' .

If $f = \{f_1, \dots, f_k\}$ in G'' is a MinCF, it must be a maximum flow; i.e., $\sum_{i=1}^k |f_i| = |f^m|$. Otherwise, it can push more flow from s to s_i to further reduce the flow's cost (as the cost of (s, s_i) is $-v_i$), contradicting that f is a MinCF. Now, to prove f 's total weight $\mathcal{V}_f = \sum_{i=1}^k (v_i \cdot |f_i|)$ in G' is the largest among all the maximum flows in G' , by way of contradiction, let's assume that there is another maximum flow $f' = \{f'_1, \dots, f'_k\}$ in G' , whose total weight $\mathcal{V}_{f'} = \sum_{i=1}^k (v_i \cdot |f'_i|) > \mathcal{V}_f$. That is, there exists a $1 \leq j \leq k$, where $f_j < f'_j$ and $f_i = f'_i$, $1 \leq i < j$. Consequently, due to that a) $|f| = |f'| = |f^m|$ (Lemma 1), b) $v_1 \geq v_2, \dots, \geq v_k$, and c) the cost of (s, s_i) is $-v_i$, the flow f' must have a smaller cost than that of f in G'' , contradicting that f is already MinCF in G'' . ■

V. APPROXIMATION ALGORITHM FOR MWF-H

We first prove that MWF-H is NP-hard, once feasible, and then propose a 2-approximation algorithm to solve it.

Theorem 4: MWF-H is NP-hard.

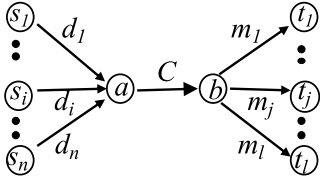


Fig. 4. MWF-H is NP-hard.

Proof: We reduce the Unbounded Knapsack Problem (UKP) [32], which is NP-hard, to MWF-H in polynomial time. UKP is defined as follows. Given k types of items, each item in type i , $1 \leq i \leq k$, has a profit p_i and a cost c_i , and a knapsack of a capacity C , its goal is to find x_i items of type i to maximize their total profit $\sum_{i=1}^k (x_i \times p_i)$ while $\sum_{i=1}^k (x_i \times c_i) \leq C$.

Given a UKP instance (k types, x_i items of type i , their profits p_i and costs c_i , and capacity C), we reduce it to an instance of MWF-H in polynomial time, as shown in Fig. 4. First, we construct the following flow network G with $k+l+2$ nodes: k source nodes and l sink nodes, and two transshipment nodes a and b . We connect s_i with a with a directed edge (s_i, a) , a with b with a directed edge (a, b) , and b with t_j with a directed edge (b, t_j) , total $k+l+1$ edges. We set edge capacities $ca(a, b)$ of (a, b) as C , $ca(s_i, a) = d_i$, source node s_i 's sending capacity, and $ca(b, t_j) = m_j$, sink node m_j 's receiving capacity.

Second, we replace x_i with $f_i(a, b)$, the number of unit flows of s_i in G , and p_i with v_i , the weight of a unit flow of s_i in G . Then, the goal of UKP to maximize $\sum_{i=1}^k (x_i \times p_i)$ s.t. $\sum_{i=1}^k (x_i \times c_i) \leq C$ becomes maximizing $\sum_{i=1}^k (f_i(a, b) \times v_i)$ subject to $\sum_{i=1}^k (f_i(a, b) \times c_i) \leq C$. Now, as (a, b) is the only edge in G linking source nodes with sink nodes, maximize $\sum_{i=1}^k (f_i(a, b) \times v_i)$ becomes the objective of the MWF-H in G with the general edge capacity constraint $\sum_{i=1}^k (f_i(a, b) \times c_i) \leq C$. Since we can reduce the UKP to MWF-H in polynomial time, MWF-H is NP-hard. ■

Approximation Algorithm. Unlike Algo. 2, Algo. 4 adopts an *all-or-nothing model*, in which each source node must either completely send its unit flows or not send at all. It executes two greedy algorithms sequentially. The first one computes the total flow weight by running Modified Edmonds-Karp on the source nodes in non-ascending order of their weights (lines 1-14), while the second does so in non-ascending order of their weight-cost ratios (lines 15-17). Lines 6-7 are the all-or-nothing model, in which the algorithm terminates if a source s_i cannot send all its d_i units of flow. Among the two computed flows f and f' , it returns the one with the larger total flow weight (lines 18-20). Its time complexity is $O(k \cdot |V'| \cdot |E'|^2)$. Below, we prove that Algo. 4 is a 2-approximation algorithm under the all-or-nothing model.

Theorem 5: In Algo. 4, $\mathcal{V}_f \geq \mathcal{V}^{opt}/2$ under the all-or-nothing model, where $\mathcal{V}^{opt}$ is the optimal total flow weight.

Proof: We refer to the two greedy algorithms in Algo. 4 as (A) and (B), each of which finds a flow with a total weight of $\mathcal{V}_f$ and $\mathcal{V}'_f$, respectively. In (B), let S_w be the set of source nodes that completely send out their unit flows, and s_j be the first source node that cannot completely send out its unit flows.

Algorithm 4 An Approximation Algorithm for MWF-H

Input: a CFN $G'(V', E')$ with S, T, d_i, v_i, c_i, m_j .

Output: flow f and its total weight $\mathcal{V}_f$.

- 1: Sort $s_i \in S$ in non-ascending order of the weight v_i . WLOG, $v_1 \geq v_2 \geq \dots \geq v_k$;
 - 2: $\forall (u, v) \in E', f(u, v) = 0, |f| = \mathcal{V}_f = 0, G'_f = G'$;
 - 3: **for** $(1 \leq i \leq k)$ **do**
 - 4: $\forall (u, v) \in E, a_i(u, v) = ca(u, v), ca(u, v) = \lfloor \frac{ca(u, v)}{c_i} \rfloor$;
 - 5: Run Algo. 1 on G' , let the max flow from s_i be f_i ;
 - 6: **if** $(|f_i| < d_i)$ **then**
 - 7: Break;
 - 8: **else**
 - 9: $|f| += |f_i|, \mathcal{V}_f += |f_i| \cdot v_i$;
 - 10: $\forall (s, s_i), ca(s, s_i) -= f_i(s, s_i)$;
 - 11: $\forall (t_j, t), ca(t_j, t) -= f_i(t_j, t)$;
 - 12: $\forall (u, v) \in E, ca(u, v) = a_i(u, v) - f_i(u, v) \cdot c_i$;
 - 13: **end if**
 - 14: **end for**
 - 15: Sort $s_i \in S$ in non-ascending order of their weight-cost ratio $\frac{v_i}{c_i}$. WLOG, $\frac{v_1}{c_1} \geq \frac{v_2}{c_2} \geq \dots \geq \frac{v_k}{c_k}$;
 - 16: $\forall (u, v) \in E', f'(u, v) = 0, |f'| = \mathcal{V}'_f = 0, G'_{f'} = G'$;
 - 17: Run lines 3-14, and let the computed flow be f' with total flow weight $\mathcal{V}'_f$;
 - 18: **if** $(|f| < |f'|)$ **then**
 - 19: $f = f', \mathcal{V}_f = \mathcal{V}'_f$;
 - 20: **end if**
 - 21: **return:** f and $\mathcal{V}_f$.
-

We have $\sum_{s_j \in S_w} (v_i \cdot d_i) = \mathcal{V}'_f$ and $v_j \cdot d_j \leq \mathcal{V}_f$, as (A) sends unit flows of maximum weight first.

In (B), if the $|f_j|$ amount of flows sent out by s_j exactly fills all the edge capacities in G' , this is an optimal solution. This is because a) (B) completely sends out all the unit flows for source nodes with weight-cost ratios $> \frac{v_i}{c_i}$, b) it fills out the remaining edge capacity in G' with unit flows of $\frac{v_i}{c_i}$, and c) and all source nodes not included have weight-cost ratios $< \frac{v_i}{c_i}$. Under the all-or-nothing model, however, the true optimum $\mathcal{V}^{opt}$, can not be larger. We have $\mathcal{V}^{opt} \leq \sum_{s_j \in S_w} (v_i \cdot d_i) + v_j \cdot |f_j| \leq \sum_{s_j \in S_w} (v_i \cdot d_i) + v_j \cdot d_j \leq \mathcal{V}'_f + \mathcal{V}_f$. As $\mathcal{V}_f = \max\{\mathcal{V}_f, \mathcal{V}'_f\}$, it must be $\mathcal{V}_f \geq \mathcal{V}^{opt}/2$. ■

We introduce the all-or-nothing model to simplify heterogeneous-capacity interactions among flows and facilitate the approximation analysis. We conjecture that approximation algorithms for the general splittable case may be obtained through LP relaxation and randomized rounding, primal-dual methods, or resource augmentation techniques. Investigating whether constant-factor approximations exist for the general MWF-H remains an interesting direction for future research.

VI. ALGORITHMIC SOLUTIONS FOR PDP

In this section, we solve PDP under both uniform and general energy models by connecting it to the solutions of MWF-U and MWF-H from Sections III to V.

BSN-based flow network (BFN). We first transform the BSN graph $G(V, E)$ into another flow network $G'(V', E')$ called *BSN-based flow network* (BFN). Compared to the capacity-restricted flow network (CFN) introduced in

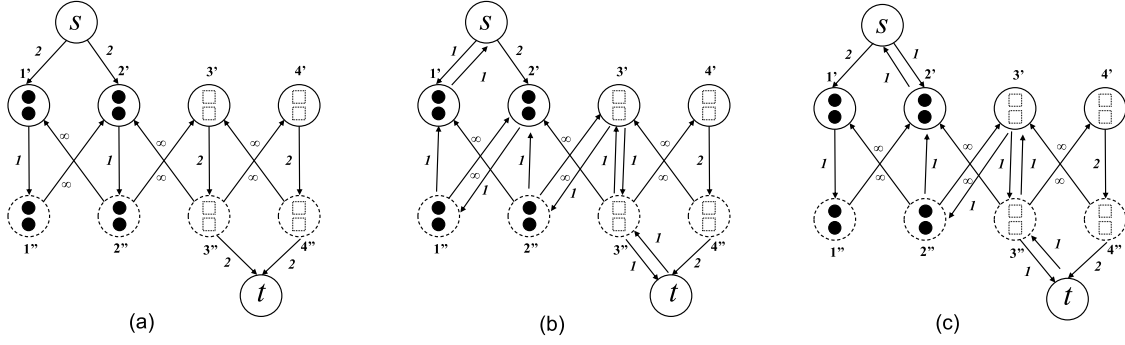


Fig. 5. (a) BSN-based flow network $G'(V', E')$ transformed from linear BSN graph in Fig. 1. (b) The residual graph after applying GOA on $G'(V', E')$. (c) The residual graph after applying the Edmonds-Karp on $G'(V', E')$.

Sections IV and V, the BFN not only converts the nodes' sending/receiving capacities into the edge capacities of G' , as CFN does, but also converts the sensor nodes' energy into edge capacities, to ensure they all play a role in finding the maximum priority in the data preservation process.

The transformation has the following four steps. Fig. 5(a) shows the BFN transformed from the linear BSN in Fig. 1.

1). Replace each undirected edge $(i, j) \in E$ with two directed edges (i, j) and (j, i) . Set their capacities to infinity.

2). Split node $i \in V$ into *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 . 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 .

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 .

A. PDP Under Uniform Energy Model

To check the feasibility of a PDP in a BSN graph $G(V, E)$, we apply the Modified Edmonds-Karp viz. Algo. 1 on the above transformed BFN $G'(V', E')$. If it is feasible, we then apply the GOA viz. Algo. 2 on $G'(V', E')$ to solve the PDP.

Theorem 6: Under the uniform energy model, GOA on $G'(V', E')$ is an optimal algorithm for PDP on $G(V, E)$.

Proof: We show that the maximum weighted flow produced by GOA on $G'(V', E')$ corresponds to data preservation with the maximum data priorities on the BSN graph $G(V, E)$. Suppose that GOA gives a net flow of P_i amount from s_i to t . This means that there are P_i amount of data items offloaded from source node s_i to a sink node in $G'(V', E')$. Because GOA is optimal (Theorem 2), $\sum_{i=1}^k (P_i \times v_i)$ is maximum among all the possible solutions. Therefore, the corresponding data preservation has the maximum amount of priority. ■

Example 7: By applying GOA on Fig. 5(a), Fig. 5(b) shows the resulting FAP: $s, 1', 1'', 2', 2'', 3', 3'',$ and t . This indicates that node 1 offloads one data item with a priority of 3. In contrast, Fig. 5(c) shows the shortest FAP found by the Modified Edmonds-Karp: $s, 2', 2'', 3', 3'',$ and t , where node 2 offloads one data item with a priority of 1. □

We present the following observation about the PDP's optimal MWF and MF solutions.

Observation 2: For a storage node, it does not relay data until its storage capacity is full in both optimal MWF and MF solutions. For a DG, however, it could relay other DGs' data items even if it has not finished preserving its own in an optimal MWF solution, whereas there always exists an MF optimal solution wherein a DG node must finish preserving all its own data items before relaying others'. □

A consequence of Observation 2 is that in an optimal PDP algorithm, data is always offloaded to the nearest storage node with available storage before being offloaded further. We call this the *cascading effect* of data preservation. This effect makes our algorithms amenable to localized distributed implementation, which we present in Section VII.

B. PDP Under General Energy Model

Feasible Condition for PDP. To find if a BSN instance is feasible, we first formulate ILP(A) to find its maximum flow.

$$\max \sum_{i \in V_d} x_{s,i'} \quad (7)$$

$$\text{s.t. } x_{s,i'} \leq d_i, i \in V_d \quad (8)$$

$$x_{i'',t} \leq m_i, i \in V_s \quad (9)$$

$$x_{s,i'} + \sum_{j'':(j'',i') \in E'} x_{j'',i'} = x_{i'',j'} = \sum_{j':(i'',j') \in E'} x_{i'',j'}, \quad (10)$$

$$\sum_{j'':(j'',i') \in E'} x_{j'',i'} = x_{i'',j'} = x_{i'',t} + \sum_{j':(i'',j') \in E'} x_{i'',j'}, \quad (11)$$

$$E_i^r \times \sum_{j'':(j'',i') \in E'} x_{j'',i'} + \sum_{j':(i'',j') \in E'} (E_i^t(j) \times x_{i'',j'}) \leq E_i, i \in V \quad (12)$$

Here, $x_{i,j}$ indicates the number of flows on edge $(i, j) \in E'$. Objective (7) maximizes the total number of preserved data items of DGs V_d . Ineq. (8) indicates the maximum number of data items DG_i can offload is d_i . Ineq. (9) indicates the maximum number of items storage node i can store is m_i . Eqns. (10) and (11) indicate the flow conservation for DGs and storage nodes, respectively. Ineq. (12) is the energy constraint of all sensor nodes. We give the theorem below without proof.

Algorithm 5 Push-Relabel (u) in Data Preservation

Input: A BSN graph $G(V, E)$ with source s and sink t ;
Output: Data-preserving flow f .
Notations: $e(u)$: node u 's excess flow;
 $h(u)$: node u 's height;
 $ca(u, w)$: residual capacity of (u, w) ;

```

1: if ( $e(u) > 0$ ) then
2:   while  $e(u) > 0$ , there exists  $(u, w)$  s.t.  $h(u) = h(w) + 1$ ,
     and  $ca(u, w) > 0$  do
3:     Push  $y = \min\{e(u), ca(u, w)\}$  through  $(u, w)$  by
     sending a message to  $w$ ;
4:      $e(u) = e(u) - y$ ;  $e(w) = e(w) + y$ ;
5:     Update  $ca(u, w)$ ;
6:   end while
7:   if ( $e(u) > 0$ ) then
8:      $h(u) = 1 + \min\{h(w) : ca(u, w) > 0\}$ ;
9:     Broadcast  $h(u)$  to neighboring nodes;
10:  end if
11: end if
12: return:  $f$ .
```

Theorem 7: In ILP(A), if $\sum_{i \in V_d} x_{s,i'} < \sum_{i=1}^k d_i = d$, then this BSN instance is feasible. ■

Solving a PDP Instance. To solve the PDP for a feasible BSN instance, we formulate another ILP viz. ILP(B) by simply replacing ILP(A)'s objective function with $\max \sum_{i \in V_d} (x_{s,i'} \cdot v_i)$, which is to maximize the total priorities preserved in the entire BSN. The constraints are the same as those of ILP(A) and thus are omitted.

VII. DISTRIBUTED ALGORITHMS FOR PDP

Our distributed framework integrates the distributed push-relabel maximum flow algorithm [25] with our priority-based algorithms presented in Sections VI and VI-B. We first review the push-relabel algorithm in the context of data preservation.

Distributed Push-Relabel Algorithm. Given a flow network $G(V, E)$ with $|V| = n$ and $|E| = m$, a source node s , and a sink node t , Algo. 5 works as follows. It sends as much flow out of s as is allowed by the edge capacities $ca(s, i')$. Then, it iteratively considers a node u with more incoming flow than outgoing flow (the difference is called *excess flow* of the node), and routes its excess flow to its neighbors. To make progress, it defines a height function $h : V \rightarrow N$ 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 u sends the extra flow to a neighbor v only if $h(u) > h(v)$. When a node can no longer send its excess flow, it increases its height and pushes the flow to the node with a lower height. A maximum flow is obtained when no additional overflowing nodes remain, except the sink and the source.

The flow of the push-relabel algorithm resembles the cascading effects observed in the data preservation process (see Observation 2). In push-relabel, flow moves through the network like water flowing downhill. In data preservation, data is always offloaded to the nearest storage before being offloaded

Algorithm 6 Distributed Algorithm for the PDP

Input: A transformed BFN $G'(V', E')$.
Output: Data preservation flow f .

```

1: Each DG broadcasts its priority to the network;
2: for (Each DG in the descending order of its priority) do
3:    $s$  pushes maximum allowable data to this DG;
4:   while (there exists a node  $u$  with positive excess) do
5:     Run Push-Relabel( $u$ ) viz. Algo. 5;
6:   end while
7: end for
8: return:  $f$ .
```

to a more distant one. Therefore, push-relabel is particularly suitable for our data preservation scheme.

Distributed Data Preservation with Data Priority. However, a few modifications to the push-relabel algorithm are needed to make the distributed data preservation work. First, the DGs need to coordinate so that higher-priority DGs push data into the network before lower-priority DGs. Second, the energy consumption of each node must be accounted for. Third, the push-relabel algorithm assumes a single source and a single sink, whereas a BSN has multiple DGs and multiple storage nodes. A transformed BFN $G'(V', E')$ proposed in Section VI is needed to address all these challenges.

The distributed algorithm, vs. Algo. 6, starts with DGs broadcasting their priorities throughout the network. Then it consists of two nested loops. It pushes the maximum allowable number of flows to the in-node of the highest-priority DG (the outer for loop), then continues the push-relabel process until no nodes with excess flow remain (the inner while loop), except the source and sink. This continues until the lowest-priority DG completes the push-relabel process, or until no more flow can be pushed into the network.

Note that the priority-based push-relabel algorithm has not been studied in networking or distributed systems. The only work that bears the name “weighted- or priority-based push-relabel” is by Bernstein et al. [6], [7]. However, the weights in the literature refer to the edge-weight function, in which edges with higher weights are relabeled less often to improve the efficiency of the push-relabel process. Thus, they are different from our priority-based push-relabel.

Theorem 8: Algo. 6 preserves maximum total priorities. It runs in $O(kn^2)$ time and uses $O(n^2m)$ messages.

Proof: Algo. 6 is a distributed implementation of the GOA. Therefore, it preserves maximum total priorities due to the optimality of the GOA (Theorem 6). It is shown in [25] 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 $|E'| = 2m + 2n$, its time and message complexities are $O(n^2)$ and $O(n^2m)$. Algo. 6 differs from the push-relabel algorithm in lines 1 and 2. Line 1 incurs $O(kn) = O(n^2)$ broadcast messages, and line 2 increases the running time by at most k times. Therefore, Algo. 6 runs in $O(kn^2)$ and uses $O(n^2m + n^2) = O(n^2m)$ messages. ■

Although the MWF time-expanded graph grows linearly with the planning horizon and may limit its applicability in highly dynamic environments, our heuristic and distributed

TABLE III
THE COMPARED ALGORITHMS

	Uniform Model	General Model	Maximum Flow
Centralized	GOA (Algo. 2), Greedy	ILP(A), Greedy	Modified Edmonds-Karp, ILP(B)
Distributed	Distributed PDP (Algo. 6)	Distributed PDP (Algo. 6)	Push Relabel [25]

TABLE IV
QUANTITATIVE RESULTS OF FIG. 6

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

algorithms avoid explicitly constructing the time-expanded network and can instead operate in an event-driven manner, making them more suitable for large-scale or time-varying BSNs. Consequently, we view the MWF formulation primarily as an offline benchmark or an upper bound for evaluating practical algorithms, rather than as a real-time deployment solution.

VIII. PERFORMANCE EVALUATION

In this section, we compare our priority-based data-preservation algorithms with classical maximum-flow and push-relabel algorithms under both uniform and general energy models, all listed in Table III. We use Google OR-Tools [1] as our ILP solver. Specifically, we use the sequential GLOP (Google Linear Optimization Package) solver, which is implemented as a primal-dual revised simplex algorithm.

Where applicable, each data point averages over 20 PDP instances, and the error bars indicate a 95% confidence interval. 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. Our implementation and code are available at: https://github.com/grivera64/PDP_MWF.

A. Uniform Energy Model

We compare our optimal GOA algorithm (**Optimal**), the greedy heuristic (**Heuristic**), the distributed algorithm (**Distributed**) with two existing algorithms that do not consider flow priorities, including the Modified Edmonds-Karp (**Edmonds-Karp**) and Push-Relabel (**PushRelabel**). Here, the greedy heuristic is a simple iterative algorithm in which, in each iteration, a DG with the highest data-priority offloads its data items to the closest storage node with available storage.

Visual Comparison in Grids. We adopt a grid for clear visual comparison. In a grid, each node can communicate directly with its (at most) four neighbors, and each edge costs 1 unit of energy. We investigate two scenarios that hinder data preservation: insufficient storage space or insufficient energy.

Insufficient Storage. Fig. 6 shows a grid of 10×10 . Four DGs, located at (4, 6), (7, 6), (4, 5), and (7, 5), are represented by different shapes, each representing a different priority. Each DG has 30 overflow data items. The remaining 96 nodes are storage nodes; each can store one data item. The initial energy level E_i of each node is 30 units. It shows that the number

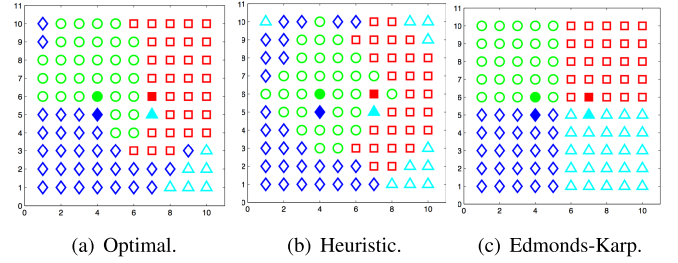


Fig. 6. Data preservation under insufficient storage. The solid circle has priority 8, the solid square 6, the solid diamond 4, and the solid triangle 2. The successfully preserved data items are represented using the corresponding hollow shapes.

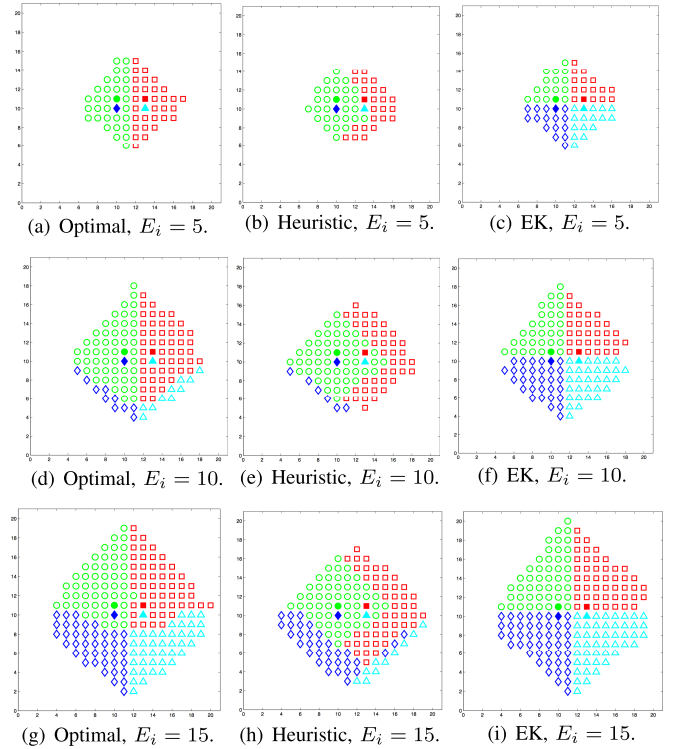


Fig. 7. Data preservation under insufficient energy with varying E_i . EK stands for Edmonds-Karp.

of preserved data items (from highest priority to lowest) is [30, 30, 30, 6] in Optimal, [30, 28, 28, 10] in Heuristic, and [24, 24, 24, 24] in Edmonds-Karp. Both Optimal and Heuristic preserve higher-priority data, whereas Optimal does a much better job of “filtering” out low-priority data. Edmonds-Karp, however, offloads the same amount of data regardless of priority, as it is priority-oblivious. Table IV includes the numerical results from Fig. 6. It shows that Optimal has the highest total preserved priority, followed by Heuristic and Edmonds-Karp, although all preserve the same number of data items due to insufficient storage.

Insufficient Energy. In a grid of 20×20 , four DGs are located at (10, 10), (13, 10), (10, 11), and (13, 11). Each DG has 50 data items to preserve. Fig. 7 visualizes the results by increasing E_i from 5 to 10 to 15. As E_i increases, the number of preserved data items increases for all three algorithms. However, Optimal and Heuristic only preserve the two data

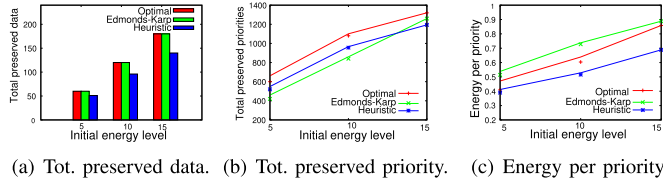


Fig. 8. Quantitative analysis of results in Fig. 7.

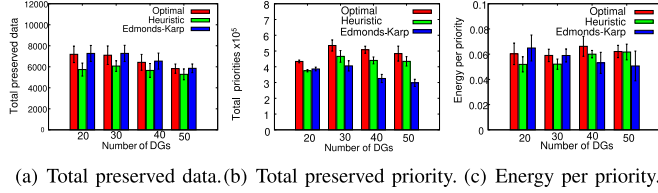


Fig. 9. Performance in general topologies with varying DGs.

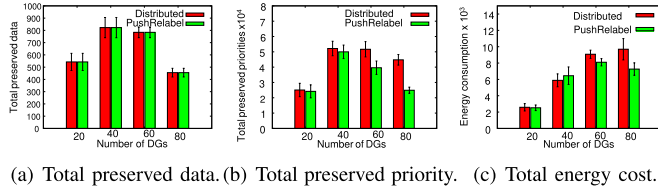


Fig. 10. Comparing Distributed and PushRelabel.

types of higher priority at $E_i = 5$, begin to preserve the other two types of lower priority at $E_i = 10$, and finally preserve a relatively large amount of the two low-priority data types at $E_i = 15$. This demonstrates the priority-preserving capabilities of these two algorithms. In contrast, Edmonds-Karp maximizes the number of preserved data items by equally preserving all four data types across varying E_i .

Fig. 8 are the quantitative results of Fig. 7. Fig. 8(a) shows that both Optimal and Edmonds-Karp preserve the largest and equal number of data items, validating our theoretical result that MWF is an MF under the uniform energy model. Fig. 8(b) shows that Optimal preserves the largest priorities. As energy efficiency is also a concern in data preservation, we define *energy per priority* as the ratio of total energy consumption to the total number of preserved priorities. Fig. 8(c) shows that Heuristics achieves the lowest energy-per-priority ratio across different energy levels, followed by Optimal and then Edmonds-Karp. This shows that Heuristics and Optimal achieve better performance and a better cost balance than Edmonds-Karp.

Distributed Algorithms. Fig. 10 compares Distributed with PushRelabel, for which we modified the distributed push-relabel algorithm implemented in [13]. It shows that both algorithms preserve the same amount of data, as both achieve maximum flow. We observe that Distributed yields a total preserved priority that is 66.7% higher than that of PushRelabel, while incurring higher energy consumption. This is because Distributed is optimal in preserving total priorities, while broadcast messages are needed to coordinate among DGs.

TABLE V
PERFORMANCE DIFFERENCE BETWEEN MWF AND MF IN PRESERVED DATA PRIORITIES BY VARYING E_i AND d_i

(a) Fig. 11: Varying E_i , $d_i = m_i = 100$	20	40	60	80	100
	12.13	8.28	5.31	2.84	0.45
(b) Fig. 12: 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

B. General Energy Model

We refer to the ILP(A) as **MF** and ILP(B) as **MWF**, respectively. We also include two greedy algorithms: one priority-based and the other priority-cost-based, denoted Greedy_p and Greedy_r, respectively. For each BSN instance, 100 sensor nodes (with a transmission range of 200m) are randomly placed in a field of 2000m × 2000m; 50 are DGs, and 50 are storage nodes. Each DG's priority is a random number in [1, 100]. Given a BSN instance, we first use ILP(A) to find the minimum E_i under which all the data items are preserved. We refer to it as the *minimum feasible energy* (MFE).

Insufficient Energy. Fig. 11 shows the performance comparison by varying E_i as 20%, 40%, 60%, and 100% of the MFE for each 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 that MWF and both greedy algorithms preserve priority more than MF, demonstrating that MF is not necessarily an MWF. Their performance difference (i.e., $\frac{\text{MWF}-\text{MF}}{\text{MF}}$) decreases with the increase of E_i , as shown in Table V(a). This is because when increasing E_i , the data-preserving capabilities of both algorithms will increase and finally converge in preserving all the data items 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 data items, while it takes another 80% of MFE to preserve the remaining 20% of the data items. This conforms to the well-known Pareto Principle [42], which says that in many societal and economic scenarios, roughly 80% of the outcomes come from 20% of causes (a.k.a. the “vital few”). Our work validates the Pareto Principle in resource allocation from a computer networking perspective.

Besides, Fig. 11(b) shows that at $E_i = 100\%$ of MFE, MF preserves more data items than MWF. This supports our finding that MWF does not necessarily achieve the maximum amount of flow. At $E_i = 100\%$ of MFE, both MF and MWF successfully preserve all 5000 data items, whereas greedy algorithms fail to do so. This can be explained by Fig. 11(c), which shows that both greedy algorithms are the least energy-efficient and MF is the most energy-efficient among the three algorithms. To investigate the tradeoff between preserved priority and incurred energy cost, Fig. 11(d) calculates the *priority-cost ratio*, which is the ratio between the total preserved priority and the total energy cost for each algorithm. We observe that the ratios decrease for all algorithms as E_i increases. This is because of the sublinear increase of preserved priority w.r.t. E_i , as shown in Fig. 11(b), and an almost linear energy cost increase w.r.t. E_i , as shown in Fig. 11(c). We also observe that MF yields a slightly higher

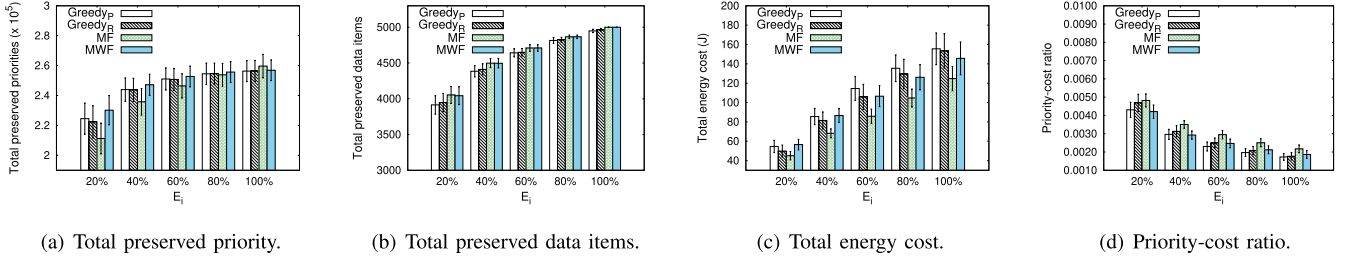


Fig. 11. Varying initial energy level E_i (in percentage of MFE) under the general energy model. Here, $d_i = m_i = 100$.

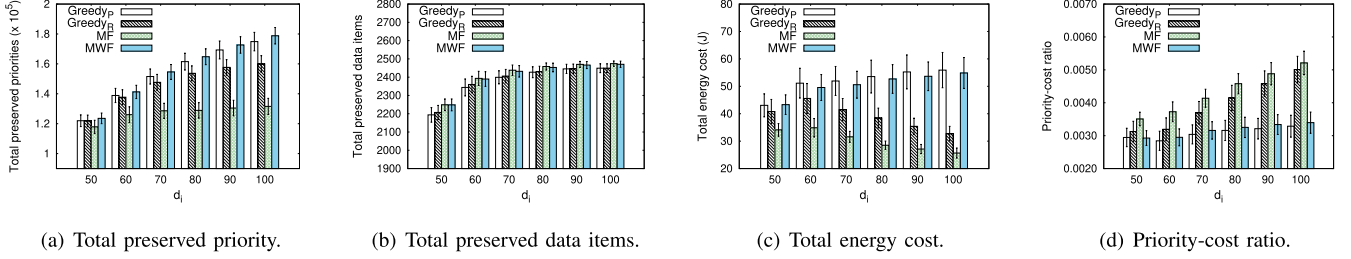


Fig. 12. Varying number of data items d_i at DGs under general energy model. Here, $m_i = 50$ and $E_i = 20\%$ of MFE.

priority-cost ratio than the other methods, mainly due to its energy-efficient shortest-path data preservation.

Insufficient Storage and Energy. We investigate data preservation in the most challenging scenario, where both energy and storage are insufficient. 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. 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, ranging from 16.78% to 33.29%, as shown in Table V(b). This shows that MWF outperforms MF in more challenging scenarios, where it can still use scarce energy and storage resources to preserve the highest-priority data items. 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 in preserved priorities for MWF is much more significant than that for MF. Despite the limited storage and energy, increasing d_i allows MWF to discard low-priority data in favor of high-priority data, thereby increasing the number of preserved priorities more quickly. In contrast, because MF is blind to data priority, increasing d_i can only increase the preserved priorities up to the limits of storage and energy. Fig. 12(c) shows that although the total energy cost of data preservation increases for MWF and Greedy_P, it generally decreases for Greedy_C, showing that Greedy_R is more energy-efficient than MWF and Greedy_P with more data packets to be preserved. Finally, Fig. 12(d) shows that MF achieves the highest priority-cost ratios. We also observe that Greedy_C yields higher priority-cost ratios than both Greedy_C and MWF, and the performance difference becomes more pronounced when d_i is large. This shows that Greedy_C strikes a better balance between preserved priorities and energy cost compared to Greedy_C and MWF.

TABLE VI
EXECUTION TIME (MS) IN BSNs WITH 100 NODES

Data Packets	Greedy	MF	MWF
60	434.1	647.6	33.3
70	426.9	836.0	36.4
80	411.8	1034.6	32.2
90	410.2	1336.3	32.6
100	414.3	1275.2	34.1

TABLE VII
EXECUTION TIME IN BSNs WITH 5000 NODES

	60	70	80	90	100
Greedy	2h5m	2h23m	2h46m	3h27m	3h18m
MF	16m1s	13m18s	16m26s	12m39s	16m58s
MWF	15m22s	17m32s	15m58s	18m38s	17m44s

Note that MWF preserved more priorities than not only MF, which is priority-oblivious, but also Greedy_P and Greedy_R, which iteratively preserve data items with the highest priorities and highest priority-cost ratio, respectively. This shows that MWF's prize-collecting gains come from its design, rather than from exploiting the priority-obliviousness of the MF baseline.

Execution Time Analysis. Table VI shows the execution time of different algorithms in Fig. 12. It shows that MWF is the fastest. This is because MWF is data priority-aware, enabling it to prune the ILP solver's search trees when two packets with different priorities require the same cost to preserve. In MF, the optimizer cannot prune the search space, increasing MF's execution time. Greedy lies between MWF and MF, as it requires constant recomputation of the best min-cost path due to dynamic energy and storage usage in data preservation.

Finally, Table VII shows the execution time in a large-scale BSN of 5000 nodes (2500 DGs and 2500 storage nodes), in a

field of $20,000\text{m} \times 20,000\text{m}$. The transmission range is 500m. We observe that both MF and MWF run very efficiently, finishing in less than 20 minutes across a range of packet numbers. This is mainly due to the search-space pruning provided by Google OR-Tools' constraint programming capability. Greedy takes two to three hours to execute, and its execution time grows linearly because it recalculates the best min-cost path at each iteration.

IX. CONCLUSION AND FUTURE WORK

This paper studied priority-based data preservation in sensing applications deployed in challenging environments. At the core are two maximum-weighted flow (MWF) problems that have not been adequately studied. Our MWF model can be summarized as multi-source and multi-sink any-to-any communication with a) capacities enforced on the nodes and b) splittable but integral unit flows with varying weights and costs. We believe our work occupies a unique niche in the large network flow landscape. As future work, we will first build a small testbed and evaluate our model using TinyOS [3]. We will focus on a simple linear or star topology to validate the assumptions and operational complexity of our algorithms.

We assume that data priorities remain fixed during a preservation interval, which allows the MWF formulations and their associated algorithms to operate on static weights. If priorities change over time, the existing algorithms can be executed periodically by updating the source weights and recomputing the preservation decisions. Since the flow network topology, energy constraints, and storage capacities remain unchanged, only the priority assignments need updating. However, the current paper provides no theoretical guarantees or experimental evaluation in this dynamic setting. Developing algorithms and approximation guarantees for time-varying priorities is, therefore, an important direction for future work.

Our data preservation and MWF model is a unique information producer-consumer model that has not yet been adequately explored. With a theoretical foundation in network flows, the techniques designed may apply to other network environments where flow priorities and resource constraints coexist. For example, modern cloud data center traffic often has different priorities and requirements, including user-generated interactive traffic, deadline-driven traffic, and long-running traffic [18], [41]. Our MWF techniques can be used to design efficient transport protocols in bandwidth-constrained cloud data centers. Another burgeoning field is large language models (LLMs) serving in GPU-based data centers [19]. Recent research [20] has formulated LLM scheduling as a maximum data-flow problem, with each unit of flow representing a token passing through a GPU node. Given that LLMs have up to hundreds of billions of parameters, some are more valuable for inference, extending MWF techniques to enable more efficient LLM scheduling in resource-constrained GPU-based data centers is an interesting new problem.

ACKNOWLEDGMENT

The authors appreciate the communication with Prof. Eva Tardos regarding the optimality of the MWF-U algorithms.

REFERENCES

- [1] *Google or Tools*. Accessed: Sep. 9, 2026. [Online]. Available: <https://developers.google.com/optimization>
- [2] *Maximum Priority Flow*. Accessed: Sep. 9, 2026. [Online]. Available: <https://www.csc.kth.se/~viggo/wwwcompendium/node120.html>
- [3] *Tinyos*. Accessed: Sep. 9, 2026. [Online]. Available: <https://github.com/tp-freeforall/prod>
- [4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
- [5] M. Bellare, "Interactive proofs and approximation: Reductions from two provers in one round," in *Proc. 2nd Isr. Symp. Theory Comput. Syst.*, 1992, pp. 266–274.
- [6] A. Bernstein, J. Blikstad, J. Li, T. Saranurak, and T.-W. Tu, "Combinatorial maximum flow via weighted push-relabel on shortcut graphs," in *Proc. 66th IEEE Symp. Found. Comput. Sci. (FOCS)*, Dec. 2025, pp. 464–485.
- [7] A. Bernstein, J. Blikstad, J. Li, T. Saranurak, and T.-W. Tu, "Maximum flow by augmenting paths in $n^{2+o(1)}$ time," in *Proc. 66th IEEE Symp. Found. Comput. Sci. (FOCS)*, May 2024, pp. 2056–2077.
- [8] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "A UAV-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling," *IEEE Access*, vol. 8, pp. 110446–110460, 2020.
- [9] I. M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *Eur. J. Oper. Res.*, vol. 88, no. 3, pp. 464–474, 1996.
- [10] C. Chekuri, M. Mydlarz, and F. B. Shepherd, "Multicommodity demand flow in a tree and packing integer programs," *ACM Trans. Algorithms*, vol. 3, no. 3, p. 27, Aug. 2007.
- [11] C.-F. Cheng and L.-H. Li, "Data gathering problem with the data importance consideration in underwater wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 78, pp. 300–312, Jan. 2017.
- [12] S. Cheng, Y. Li, Z. Tian, W. Cheng, and X. Cheng, "A model for integrating heterogeneous sensory data in IoT systems," *Comput. Netw.*, vol. 150, pp. 1–14, Feb. 2019.
- [13] B. Cherkassky and A. Goldberg, "On implementing the push—Relabel method for the maximum flow problem," *Algorithmica*, vol. 19, pp. 390–410, Apr. 1997.
- [14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.
- [15] R. I. D. Silva, J. D. C. V. Rezende, and M. J. F. Souza, "Collecting large volume data from wireless sensor network by drone," *Ad Hoc Netw.*, vol. 138, Jan. 2023, Art. no. 103017.
- [16] A. Eid, J. C. Rademacher, W. Akbar, P. Wang, A. Allam, and F. Adib, "Enabling long-range underwater backscatter via Van Atta acoustic networks," in *Proc. ACM SIGCOMM*, 2023, pp. 1–19.
- [17] L. Terray et al., "From sensor to cloud: An IoT network of radon outdoor probes to monitor active volcanoes," *Sensors*, vol. 20, no. 10, p. 2755, May 2020.
- [18] M. N. Pathan et al., "Priority based energy and load aware routing algorithms for SDN enabled data center network," *Comput. Netw.*, vol. 240, Feb. 2024, Art. no. 110166.
- [19] Q. Hu et al., "Characterization of large language model development in the datacenter," in *Proc. NSDI*, 2025, pp. 709–729.
- [20] Y. Mei, Y. Zhuang, X. Miao, J. Yang, Z. Jia, and R. Vinayak, "Helix: Serving large language models over heterogeneous GPUs and network via max-flow," in *Proc. 30th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2025, pp. 586–602.
- [21] P. Fatourou, M. Mavronicolas, and P. Spirakis, "Max-min fair flow control sensitive to priorities," in *Proc. 2nd Int. Conf. Princ. Distrib. Syst.*, 1999, pp. 45–59.
- [22] S. Ganeriwal, L. K. Balzano, and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," *ACM Trans. Sensor Netw.*, vol. 4, no. 3, pp. 1–37, Jun. 2008.
- [23] N. Garg, V. V. Vazirani, and M. Yannakakis, "Primal-dual approximation algorithms for integral flow and multicut in trees," *Algorithmica*, vol. 18, no. 1, pp. 3–20, May 1997.
- [24] P. Gjanci, C. Petrioli, S. Basagni, C. A. Phillips, L. Bölöni, and D. Turgut, "Path finding for maximum value of information in multi-modal underwater wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 404–418, Feb. 2018.
- [25] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proc. 18th Annu. ACM Symp. Theory Comput. - STOC*, 1986, pp. 136–146.
- [26] S. Guo, H. Zhang, Z. Zhong, J. Chen, Q. Cao, and T. He, "Detecting faulty nodes with data errors for wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 3, pp. 1–27, May 2014.

- [27] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, pp. 1–10.
- [28] X. Hou, Z. Sumpter, L. Burson, X. Xue, and B. Tang, "Maximizing data preservation in intermittently connected sensor networks," in *Proc. IEEE 9th Int. Conf. Mobile Ad-Hoc Sensor Syst. (MASS)*, Oct. 2012, pp. 448–452.
- [29] S. Hsu, Y. Yu, and B. Tang, "DRE2: Achieving data resilience in wireless sensor networks: A quadratic programming approach," in *Proc. IEEE 17th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Dec. 2020, pp. 71–79.
- [30] H. Huang, A. V. Savkin, M. Ding, and C. Huang, "Mobile robots in wireless sensor networks: A survey on tasks," *Comput. Netw.*, vol. 148, pp. 1–19, Jan. 2019.
- [31] K. Jansen and S. E. J. Kraft, "A faster FPTAS for the unbounded knapsack problem," *Eur. J. Combinatorics*, vol. 68, pp. 148–174, Feb. 2018.
- [32] H. Kim and S.-G. Min, "Priority-based QoS MAC protocol for wireless sensor networks," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, May 2009, pp. 1–8.
- [33] J. Könemann, O. Parekh, and D. Pritchard, "Max-weight integral multicommodity flow in spiders and high-capacity trees," in *Proc. Int. Workshop Approximation Online Algorithms*, 2009, pp. 1–14.
- [34] D. Kozen, "Lexicographic flow," Dept. Comput. Inf. Sci., Cornell Univ., Tech. Rep., Jun. 2009. [Online]. Available: <http://hdl.handle.net/1813/13018>
- [35] R. Kumar et al., "Mitigating performance degradation in congested sensor networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 6, pp. 682–697, Jun. 2008.
- [36] Y. Lai et al., "Identifying malicious nodes in wireless sensor networks based on correlation detection," *Comput. Secur.*, vol. 113, Feb. 2022, Art. no. 102540.
- [37] K. Li, W. Ni, and F. Dressler, "Continuous maneuver control and data capture scheduling of autonomous drone in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2732–2744, Aug. 2022.
- [38] W. Liang, Z. Xu, W. Xu, J. Shi, G. Mao, and S. K. Das, "Approximation algorithms for charging reward maximization in rechargeable sensor networks via a mobile charger," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3161–3174, Oct. 2017.
- [39] Q. Liu, Z. Wang, X. He, and D. H. Zhou, "On Kalman-consensus filtering with random link failures over sensor networks," *IEEE Trans. Autom. Control*, vol. 63, no. 8, pp. 2701–2708, Aug. 2018.
- [40] S. Marti, T. J. Giuli, K. Lain, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. MobiCom*, 2000, pp. 255–265.
- [41] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1492–1525, 2nd Quart., 2018.
- [42] V. Pareto. *Pareto Principle*. Accessed: Sep. 9, 2026. [Online]. Available: https://en.wikipedia.org/wiki/Pareto_principle
- [43] 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. IEEE ICCCN*, Jul. 2017, pp. 1–9.
- [44] J. Rivera, Y. Chen, and B. Tang, "On the performance of Nash equilibria of data preservation in base station-less sensor networks," in *Proc. IEEE MASS*, Sep. 2023, pp. 252–260.
- [45] W. Rothfarb, N. P. Shein, and I. T. Frisch, "Common terminal multicommodity flow," *Oper. Res.*, vol. 16, no. 1, pp. 202–205, 1968.
- [46] K. Salimifard and S. Bigharaz, "The multicommodity network flow problem: State of the art classification, applications, and solution methods," *Oper. Res.*, vol. 22, no. 1, pp. 1–47, Mar. 2022.
- [47] D. Wohwe Sambo and A. Förster, "Wireless underground sensor networks: A comprehensive survey and tutorial," *ACM Comput. Surveys*, vol. 56, no. 4, pp. 1–44, Oct. 2023.
- [48] K. Saxena, N. Gupta, J. Gupta, D. K. Sharma, and K. Dev, "Trajectory optimization for the UAV assisted data collection in wireless sensor networks," *Wireless Netw.*, vol. 28, no. 4, pp. 1785–1796, May 2022.
- [49] D. Sun, J. Wu, J. Yang, and H. Wu, "Intelligent data collaboration in heterogeneous-device IoT platforms," *ACM Trans. Sensor Netw.*, vol. 17, no. 3, pp. 1–17, Jun. 2021.
- [50] B. Tang, "DAO²: Overcoming overall storage overflow in intermittently connected sensor networks," in *Proc. IEEE INFOCOM*, Jul. 2018, pp. 135–143.
- [51] B. Tang, N. Jaggi, H. Wu, and R. Kurkal, "Energy-efficient data redistribution in sensor networks," *ACM Trans. Sensor Netw.*, vol. 9, no. 2, pp. 1–28, Mar. 2013.
- [52] B. Tang, H. Ngo, Y. Ma, and B. Alhakami, "DAO²: Overcoming overall storage overflow in intermittently connected sensor networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 3143–3158, Jun. 2023.
- [53] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *Eur. J. Oper. Res.*, vol. 255, no. 2, pp. 315–332, Dec. 2016.
- [54] C. Wang, J. Li, Y. Yang, and F. Ye, "Combining solar energy harvesting with wireless charging for hybrid wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 3, pp. 560–576, Mar. 2018.
- [55] A. Wichmann, T. Korkmaz, and A. S. Tosun, "Robot control strategies for task allocation with connectivity constraints in wireless sensor and robot networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 6, pp. 1429–1441, Jun. 2018.
- [56] Q. Wu, P. Sun, and A. Boukerche, "Unmanned aerial vehicle-assisted energy-efficient data collection scheme for sustainable wireless sensor networks," *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106927.
- [57] W. Xu et al., "Throughput maximization of UAV networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 881–895, Apr. 2022.
- [58] W. Xu et al., "Reward maximization for disaster zone monitoring with heterogeneous UAVs," *IEEE/ACM Trans. Netw.*, vol. 32, no. 1, pp. 890–903, Feb. 2024.
- [59] W. Xu et al., "Approximation algorithms for the generalized team orienteering problem and its applications," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 176–189, Feb. 2021.
- [60] W. Xu et al., "Minimizing the deployment cost of UAVs for delay-sensitive data collection in IoT networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 812–825, Apr. 2022.
- [61] W. Xu et al., "Approximation algorithms for the team orienteering problem," in *Proc. IEEE INFOCOM*, Jul. 2020, pp. 1389–1398.
- [62] X. Xue, X. Hou, B. Tang, and R. Bagai, "Data preservation in intermittently connected sensor networks with data priority," in *Proc. IEEE Int. Conf. Sens., Commun. Netw. (SECON)*, Jun. 2013, pp. 122–130.
- [63] 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. Sensor Netw.*, vol. 20, no. 1, pp. 1–40, Oct. 2023.
- [64] Z. Yuan, K. Liu, Z. Xu, H. Wang, Y. Liu, and X. Tang, "Development of micro-radioisotope thermoelectric power supply for deep space exploration distributed wireless sensor network," *Adv. Astronaut. Sci. Technol.*, vol. 3, no. 2, pp. 157–163, Dec. 2020.
- [65] J. Zhang et al., "Minimizing the number of deployed UAVs for delay-bounded data collection of IoT devices," in *Proc. IEEE INFOCOM*, May 2021, pp. 1–10.
- [66] Z. Zhao, C. Liu, W. Qu, and T. Yu, "An energy efficiency multi-level transmission strategy based on underwater multimodal communication in UWSNs," in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1579–1587.