A Generalized Data Preservation Problem in Sensor Networks - A Network Flow Perspective

Bin Tang[1], Rajiv Bagai[2], FNU Nilofar[2], and Mehmet Bayram Yildirim[3]

[1] Dept. of Computer Science, California State Univ., Dominguez Hills, USA
[2] Dept. of Electrical Engineering and Computer Science, Wichita State Univ., USA
[3] Dept. of Industrial and Manufacturing Engineering, Wichita State Univ., USA
btang@csudh.edu, rajiv.bagai@wichita.edu, nilu23@gmail.com,
bayram.yildirim@wichita.edu

Abstract. Many emerging sensor network applications require sensor node deployment in challenging environments that are remote and inaccessible. In such applications, it is not always possible to deploy base stations in or near the sensor field to collect sensory data. Therefore, the overflow data generated by some nodes is first offloaded to other nodes inside the network to be preserved, then gets collected when uploading opportunities become available. In this paper, we study a *generalized data preservation problem* in sensor networks, whose goal is to minimize the total energy consumption of preserving data inside sensor networks, given that each node has limited battery power. With an intricate transformation of the sensor network graph, we demonstrate that this problem can be modeled and solved as a minimum cost flow problem. Also, using data preservation in sensor networks as an example, we show that seemingly equivalent maximum flow techniques can result in dramatically different network performance. Much caution thus needs to be exercised while adopting classic network flow techniques into sensor network applications, despite successful application of network flow theory to many existing sensor network problems. Finally, we present a load-balancing data preservation algorithm, which not only minimizes the total energy consumption, but also maximizes the minimum remaining energy of nodes that receive distributed data, thereby preserving data for longer time. Simulation results show that compared to the existing techniques, this results in much evenly distributed remaining energy among sensor nodes.

Keywords: Data Preservation, Network Flow, Sensor Networks

1 Introduction

Data preservation is critical in sensor networks that are deployed in challenging environments, such as underwater or ocean sensor networks [1, 2], acoustic sensor networks [3], and sensor networks monitoring volcano eruption and glacial melting [4, 5]. Due to limited accessibility in these environments, it is not possible to deploy a base station with power outlet near or inside the sensor network to collect the data. Meanwhile, each sensor node has limited storage capacity

and a finite battery supply. Sensor nodes close to the event of interest constantly generate large amounts of sensory data, which can quickly exhaust their limited storage capacity. We refer to these sensor nodes with exhausted data storage as *source nodes*. Other sensor nodes that still have available storage are referred to as *destination nodes*. In order to prevent data loss, the overflow data generated at source nodes needs to be offloaded to some destination nodes before uploading opportunities (such as data mules [6, 7] or low rate satellite link [8]) become available. We refer to this process as *data preservation in sensor networks*.

Previous research on data preservation [9] has two major limitations. First, it assumes that each node has "enough" battery power, so data items can always be offloaded from source to destination nodes using shortest paths (in terms of number of hops) between them. In this paper, we study a more challenging and general problem, wherein each node has limited battery power, therefore shortest paths may not always be viable. By defining and solving a *generalized data preservation problem*, we demonstrate that even with low energy levels of sensor nodes, optimal data preservation is still achievable. In particular, by fine-tuning the costs and capacities of edges of the flow network transformed from the sensor network graph, we show that the generalized data preservation problem is equivalent to the minimum cost flow problem, which can be solved optimally and efficiently [10]. Second, works such as [9] only focus on total energy consumption in data preservation and do not pay attention to load-balancing of energy consumption of individual sensor nodes. Once a node storing data items depletes its energy, the data preservation fails. Therefore, maintaining load-balancing among sensor nodes is critical for data preservation. We achieve load-balancing by maximizing the minimum remaining energy among all destination nodes. In contrast, Hou et al. [11] do not minimize the total energy consumption of data preservation, while Patel et al. [12] provide separate solutions for minimizing the routing cost and maximizing the minimum remaining energy.

Network flow theory [10] has been adopted to solve many fundamental problems in sensor networks, including data gathering [12–14], data aggregation [15], and clustering [16]. Classic network flow problems including maximum flow, minimum cost flow and multi-commodity flow have all been employed in sensor network research (Section 2 contains a review of such work). Using data preservation as an example, however, we demonstrate that much caution needs to be exercised while adopting classic maximum flow techniques into sensor network applications, as seemingly equivalent techniques can result in dramatically different network performance. In particular, we show that different maximum flow algorithms, viz. the Ford-Fulkerson Algorithm and the Edmonds-Karp Algorithm [10], which differ only in time complexity, yield dramatically different energy consumption in sensor networks.[1]

We also empirically compare and analyze the performance of our chosen maximum flow algorithm (i.e., Edmonds-Karp Algorithm) and minimum cost flow algorithm, for various network scenarios. Based on simulation results, we draw

[1] We focus here only on the data preservation problem in sensor networks [9, 11, 17], but our findings are applicable to many other sensor network applications as well.

some conclusions as to what extent and how well the network flow algorithms can be applied to solve sensor network problems.

Paper Organization. The rest of the paper is organized as follows. In Section 2 we give an overview of sensor network research that adopts network flow algorithms. In Section 3 we formulate and solve the generalized data preservation problem. We also solve a related problem that finds the maximum number of data items that can be offloaded, and show that two classic maximum flow algorithms yield very different performance. Section 4 proposes load-balancing data preservation algorithm. Section 5 gives some analysis of the simulation results. We conclude the paper in Section 6 and discuss possible future work.

2 Related Work

The network flow algorithms adopted in sensor network research include maximum flow [11, 13, 14, 17], minimum cost flow [9, 12, 18], and multi-commodity flow [15]. Below we give a brief review.

Maximum Flow Problem: Hong et al. [14] study store-and-gather problems in sensor networks, and show that these are essentially flow maximization under vertex capacity constraint, which reduces to a standard maximum flow problem. Bodlaender et al. [13] study integer maximum flow in wireless sensor networks with energy constraint. They show that despite the efficiency of traditional maximum flow methods, integer maximum flow in sensor networks is indeed strongly NP-complete and in fact APX-hard [19], which means it is unlikely to have a polynomial time approximation scheme. Xue et al. [17] let sensory data from different sensor nodes have different priorities, and study how to preserve data inside the network with highest priorities. They model the problem as a maximum weighted flow problem, wherein different flows have different weights. Hou et al. [11] study the data preservation problem in intermittently connected sensor networks and design a maximum flow based algorithm to maximize data preservation time in the network. Besides, they observe that due to energy constraints at sensor nodes, it is possible that not all overflow data items can be preserved. They propose a Modified Edmonds-Karp Algorithm to find out if this is the case. We show that with more intricate transformation of the flow network, Edmonds-Karp Algorithm can be applied directly without being modified.

Minimum Cost Flow Problem: Patel et al. [12] minimize the energy cost of sending data packets from sensor nodes to base stations while satisfying the capacity limits of wireless links, and propose a routing protocol based on the minimum cost flow algorithm. Ghiasi et al. [16] study a so called balanced k-clustering problem in sensor networks, wherein each of the k clusters is balanced (in terms of number of sensor nodes) and the total distance between sensor nodes and master nodes is minimized. They show that the k-clustering problem can be modeled as a minimum cost flow problem. Tang et al. [9] formulate the energy-efficient data redistribution problem in data-intensive sensor networks as a minimum cost flow problem and present a distributed algorithm.

Multi-commodity Flow Problem: Xue et al. [15] study energy efficient routing for data aggregation in wireless sensor networks, with the goal of maximizing the lifetime of the network. The resulting model is a multi-commodity flow problem, where each commodity represents the data generated from a sensor node. Since multi-commodity flow problem is NP-hard, they propose a fast ϵ-approximation algorithm, and extend their algorithm for multiple base stations.

In contrast to existing research, our work takes a new perspective by studying how different network flow algorithms could have different effect on sensor network performance such as energy consumption. We believe this is an important effort – as shown in Section 3.2, network flow modeling does not necessarily take into account resource consumption/allocation in a network-specific context.

3 Generalized Data Preservation Problem

3.1 System Model and Problem Formulation.

System Model. We model the sensor network as an undirected graph $G(V, E)$, where $V = \{1, 2, ..., |V|\}$ is the set of $|V|$ nodes, and E is the set of $|E|$ edges. There are p source nodes, denoted as V_s. Without loss of generality, let $V_s = \{1, 2, ..., p\}$. Source node i is referred to as SN i. Let d_i denote the number of overflow data items SN i needs to offload. Let $q = \sum_{i=1}^{p} d_i$ be the total number of data items to be offloaded in the network. Let c_i be the available free storage space (in terms of number of data items) at sensor node $i \in V$. Note that a source node does not have available storage space.

Sensor node i has a finite and unreplenishable initial energy E_i. We adopt first order radio model [20] as the energy model for wireless communication. In this model, for R-bit data over distance l, the transmission energy $E_t(R, l) = E_{elec} \times R + \epsilon_{amp} \times R \times l^2$, and the receiving energy $E_r(R) = E_{elec} \times R$. E_{elec} is the energy consumption per bit on the transmitter circuit and receiver circuit, and ϵ_{amp} calculates the energy consumption per bit on the transmit amplifier. For densely and uniformly deployed sensor nodes, we can approximate that $E_t(R, l) = E_r(R)$. To be consistent with the assumption in [9] that energy consumption of sending a data item from a source node to a destination node equals the number of hops it traverses, we assume that for each node, sending or receiving a data item each costs 0.5 unit of its energy.

Problem Formulation. Let $D = \{D_1, D_2, ..., D_q\}$ denote the set of q data items to be offloaded in the entire network. Let $S(i) \in V_s$, where $1 \leq i \leq q$, denote the *source node* of data item D_i. An *offloading function* is defined as $r : D \to V - V_s$, indicating $D_i \in D$ is offloaded from $S(i)$ to its *destination node* $r(i) \in V - V_s$. Let $P_i : S(i), ..., r(i)$, referred to as the *offloading path* of D_i, be the sequence of distinct sensor nodes along which D_i is offloaded from $S(i)$ to $r(i)$ (note that the offloading path is not necessarily the shortest path between source node and destination node). Let $\mathcal{E}_i$ be the energy consumption of offloading D_i from $S(i)$ to $r(i)$ following P_i ($\mathcal{E}_i$ equals the number of nodes on P_i minus one). Let E'_i denote i's energy level after all the q data items are offloaded.

The objective of generalized data preservation problem is to find an offloading scheme r and a set of paths $\mathcal{P} = \{P_1, P_2, ..., P_q\}$, to send each of the q data items to its destination node, such that the total energy consumption in this process is minimized, i.e. $\min_{r,\mathcal{P}} \sum_{1 \le i \le q} \mathcal{E}_i$, under the energy constraint: $E'_j \ge 0, \forall\ j \in V$, and the storage capacity constraint: $|\{i \mid r(i) = j, 1 \le i \le q\}| \le c_j, \forall\ j \in V$.

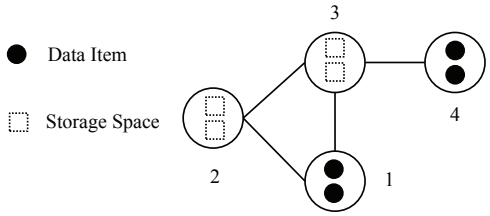


Fig. 1. A small sensor network of 4 nodes.

Example 1. Fig. 1 gives an example of the generalized data preservation problem in a small sensor network of 4 nodes. Nodes 1 and 4 are source nodes, with 2 and 2 data items to offload, respectively. Nodes 2 and 3 are destination nodes, with 2 and 2 available storage spaces, respectively. The initial energy of each node is 10. The optimal solution is that node 1 offloads its two data items to node 2, while node 4 offloads its two data items to node 3, resulting in minimum total energy consumption of 4. Other solutions are non-optimal.

3.2 Maximum Flow Algorithms to Determine the Maximum Number of Offloaded Data Items.

If the energy levels get low, not all the overflow data items at source nodes can be offloaded. Therefore, a related question is: *What is the maximum number of data items that can be offloaded given that the energy levels of sensor nodes are low?* For example, in the sensor network of Fig. 1, if the initial energy level of each node is 0.5 (instead of 10), source node 1 and 4 can each only offload 1 data item, and destination node 2 and 3 can each receive and store 1 data item.

Lemma 1 *In an optimal solution that maximizes number of offloaded data items, a SN does not relay data unless it finishes offloading all its own data items, a destination node does not relay data unless its own storage is full.*

Proof: By way of contradiction, assume that in the optimal solution, there is a SN B that serves as a relaying node before finishing offloading all its data items. That is, there exists in the optimal solution an offloading path $P : A, ..., B, ..., C$, which offloads the overflow data items of SN A to destination node C, while SN B still has its own overflow data items to offload. Assume that a data items are offloaded from A, and B still has b amounts of data items to offload. We therefore can select the path $P' : B, ..., C$, along which B offloads $\min(a, b)$ data

items to C (A offloads $\max(0, a-b)$ data items to C along P). This strategy achieves the same maximum amount of offloaded data, while costing less energy than the optimal solution. The argument for destination nodes is similar. □

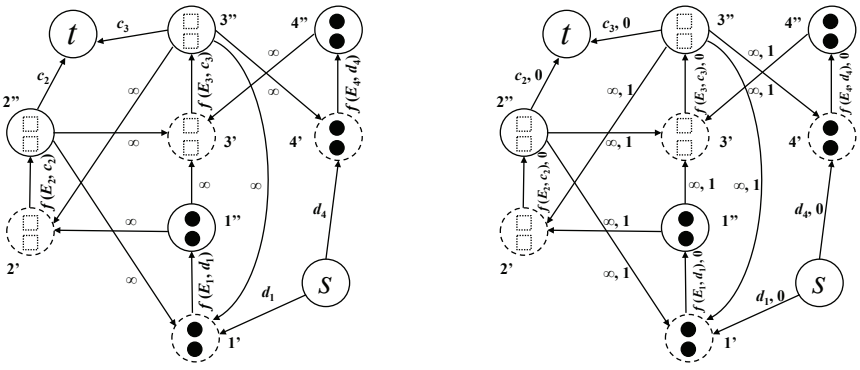
To find the maximum amount of data items offloaded, we first transform the undirected graph $G(V, E)$ into a new directed graph $G'(V', E')$ as follows:

I. $V' = V \cup \{s, t\}$, where s is the new source node and t is the new sink node.
II. Replace each undirected edge $(i, j) \in E$ with two directed edges (i, j) and (j, i). Set their edge capacities as infinity.
III. Split each node $i \in V$ into two nodes: *in-node* i' and *out-node* i''. All incoming directed edges of i are incident on i' and all outgoing directed edges of i emanate from i''. The edge capacity of (i', i'') is $f(E_i, d_i)$ for source node SN i, and $f(E_i, c_i)$ for destination node i. $f(x, y)$ is defined as:

$$f(x, y) = \begin{cases} 2x & \text{if } (x < y/2), \\ x + y/2 & \text{otherwise.} \end{cases} \quad (1)$$

IV. Connect s to in-node of SN $i \in V_s$ with an edge of capacity d_i. Connect out-node of destination node $j \in V - V_s$ to t with an edge of capacity c_j.

Therefore $|V'| = 2|V| + 2$ and $|E'| = 2|E| + 2|V|$. Fig. 2(a) shows the transformed graph G' of the sensor network in Fig. 1.



(a) Maximum flow problem.

(b) Minimum cost flow problem.

Fig. 2. The transformed graphs of the sensor network in Fig. 1.

Theorem 1 *Finding the maximum number of offloaded data items in $G(V, E)$ is equivalent to finding the maximum flow in $G'(V', E')$.*

Proof: First we explain the rationale for $f(x, y)$ in Equation 1. We focus on source nodes, but the same analysis works also for destination nodes. According

to Lemma 1, each source node offloads its own data items before relaying data items for others. Specifically, for SN i,

- when $E_i < d_i/2$, SN i does not have enough energy to offload all its d_i data items, since it needs 0.5 unit of energy for each data item. But in Fig. 2(a), each unit of flow reduces one unit of edge capacity, signifying one unit of energy cost for SN i. Therefore the edge capacity of (i', i'') in Fig. 2(a) is set as $2 \times E_i$, which is the largest amount of offloaded data items allowed by E_i.
- when $E_i \geq d_i/2$, SN i has enough energy to offload all its d_i data items. We set the edge capacity of (i', i'') in Fig. 2(a) as $E_i + d_i/2$. If it does offload all d_i data items, the edge capacity of (i', i'') becomes $E_i + d_i/2 - d_i = E_i - d_i/2$, which is exactly SN i's energy after it offloads all its d_i data items. Otherwise,[2] since it will not serve as relaying nodes either according to Lemma 1, adding $d_i/2$ upon E_i does not increase SN i's ability to offload more data.

Now if the value of the maximum flow from s to t in Fig. 2(a) is f, with f_i amount of flow on edge (s, i') and $\sum_{i=1}^{p} f_i = f$, there must be f_i amount of net flow out of SN i, meaning SN i offloads f_i amount of its own data items. On the other hand, if SN i can offload its f_i number of data items ($f_i \leq d_i$) following an offloading path P_i from SN i to a destination node, then in Fig. 2(a), it can offload f_i units of flow from s to t, without violating the capacity conditions of edges, giving maximum $\sum_{i=1}^{p} f_i$ amount of flow. □

However, above maximum flow modelling does not fully address data preservation from the efficient resource allocation perspective. As we will see next, different maximum flow algorithms could result in very different energy consumption in sensor networks.

Comparing Ford-Fulkerson and Edmonds-Karp. Both Ford-Fulkerson and Edmonds-Karp are classic maximum flow algorithms. In each iteration of Ford-Fulkerson, an arbitrary augmenting path is selected in the residual graph to push flow from source to sink, whereas in Edmonds-Karp, a shortest augmenting path is selected. The time complexity of Ford-Fulkerson is $O(|E'|C)$, where C is the value of a maximum flow in G'. The time complexity of Edmonds-Karp is $O(|V'||E'|^2)$. Note that Hou et al. [11] designed a modified Edmonds-Karp maximum flow algorithm, called MEA, to determine the maximum number of data items offloaded. Our findings are an improvement upon theirs. First, Theorem 1 shows that with intricate specification of the edge capacity of the transformed graph (i.e., Equation 1), any maximum flow algorithm can be directly applied to the transformed graph without modification. More fundamentally, we observe that when being applied to solve sensor network problems, different classic maximum flow algorithms, namely Ford-Fulkerson algorithm and Edmonds-Karp algorithm, could result in very different energy consumption, even though both achieve maximum amount of flow and only differ in time complexity. This is not explored in [11].

[2] Note that this is possible when destination nodes around SN i do not have enough energy to store all d_i data items.

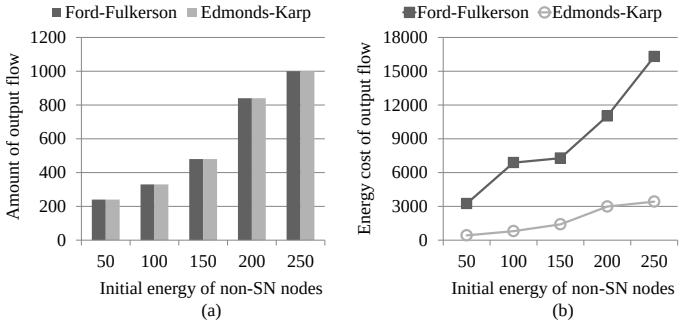


Fig. 3. Comparison between the Ford-Fulkerson and the Edmonds-Karp algorithms of the flow amounts and costs of their output flows for 1000 data items at SN nodes, over different initial energy levels of non-SN nodes.

To illustrate, we create a 10×10 grid network (with 100 nodes). One node is randomly selected as the source node with 1000 data items to offload. For each destination node, its storage capacity is 20, and its initial energy levels is varied as 50, 100, 150, 200, 250.[3] Fig. 3(a) shows that both algorithms offload the same number of data items because both achieve maximum flow. However, Fig. 3(b) shows that Edmonds-Karp costs much less energy than Ford-Fulkerson does, and the difference gets larger with the increase of energy levels.

3.3 Minimum Cost Flow Algorithm.

The minimum cost flow problem [10] is the following: Given a graph in which each edge has a capacity and a cost, some nodes are supply nodes and some are demand nodes, and total supply equals total demand; the goal is to find flows from supply nodes to demand nodes with minimum total cost while the capacity constraint at each edge is satisfied. To find the optimal algorithm for generalized data preservation, we first transform undirected graph $G(V, E)$ into another new directed graph $G''(V'', E'')$. Much of the transformation is the same as the one in Section 3.2. Additionally, for any edge (i, j) in G, we set the costs of corresponding edges (i'', j') and (j'', i') in G'' to be 1. We set the costs of all other edges in G'' as 0. Finally, we set both the supply at s and the demand at t as $\sum_{i=1}^{p} d_i$, the total number of data items to be offloaded in the entire network. Fig. 2(b) shows the transformed network graph G'' corresponding to the sensor network in Fig. 1.

Theorem 2 *The generalized data preservation problem in $G(V, E)$ is equivalent to the minimum cost flow problem in $G''(V'', E'')$.*

[3] We assign the source node large enough energy so that all 1000 data items can be offloaded. Otherwise, the amount of data offloaded in the network (i.e., the effect of maximum flow algorithms) is mainly limited by the energy level of source node, making the comparison less interesting.

Proof: We show that a minimum cost flow from s to t in G'' solves the generalized data preservation problem in G optimally. Specifically, we show that, a) it offloads all the data items from their source nodes to some destination nodes, and b) it incurs minimum energy cost in this process.

A minimum cost flow from s to t must include d_i amount of flow on edge (s, i') in G'' $(1 \leq i \leq p)$, since the capacity of (s, i') is d_i and the total amount of flow from s to t is $\sum_{i=1}^{p} d_i$. This signifies that in G, d_i amount of data items are offloaded from SN i. For any data item D_i in G, its corresponding flow in G'' goes from s to $S(i)'$, $S(i)''$, ..., $r(i)'$, $r(i)''$, and ends at t, indicating that D_i is finally stored at destination node $r(i)$. Besides, the capacity of edge (i'', t) being c_i, the storage capacity of destination node i, guarantees that in G, a destination node never stores more than its storage capacity allows.

For an edge (i, j) in G, sending a data item between i and j costs one amount of energy, which is accurately captured in G'', wherein the costs of corresponding edges (i'', j') and (j'', i') are 1 while costs of others are all 0. The minimum cost of sending $\sum_{i=1}^{p} d_i$ amount of flow from s to t in G'' is therefore incurring minimum amount of energy cost offloading d_i amount of data from SN i $(1 \leq i \leq p)$. □

Time Complexity. There are various polynomial algorithms to solve minimum cost flow problem. In this paper, we use the algorithm and implementation by Goldberg [21, 22] due to its practical nature. This algorithm has the time complexity of $O(|V''|^2|E''|log(|V''|\mathcal{C}))$, where $|V''|$, $|E''|$, and $\mathcal{C}$ are, respectively, the number of nodes, edges, and the maximum capacity of an edge in graph G''. Since $|V''| = 2|V| + 2$ and $|E''| = 2|E| + 2|V|$, the time complexity of the minimum cost flow algorithm is therefore $O(|V|^2|E|log(|V|\mathcal{C}))$.

4 Data Preservation Problem With Load Balancing

Problem Formulation. The goal of data preservation problem with load-balancing is first to minimize the total energy consumption in data preservation; then among the minimum total energy consumption solutions, to find the one that maximizes the minimum remaining energy among all the destination nodes. Specifically, we find an offloading scheme r and a set of paths $\mathcal{P} = \{P_1, P_2, ..., P_q\}$, to offload each of the q data items to its destination node, such that the total energy consumption in this process is minimized, i.e. $\min_{r,\mathcal{P}} \sum_{1 \leq i \leq q} \mathcal{E}_i$, and the minimum remaining energy among all the destination nodes is maximized, i.e., $\max_{r,\mathcal{P}} \min_{1 \leq i \leq q} E'_{r(i)}$.

Finding All Shortest Paths Between Nodes In Grid Networks. Before presenting the load-balancing algorithm (Algorithm 2), we first find all the shortest paths between any given two nodes in a grid network (Algorithm 1). There is extensive research on finding the k shortest simple paths in a directed weighted graph [23, 24]. In this paper we use a grid network for clarity of presentation, and design a much simpler recursive algorithm.[4] The algorithm works as follows.

[4] However, we are aware of that [23, 24] present much more efficient algorithms using complex data structures, which are difficult to implement.

In the base case, when the source and destination nodes are the same, it returns only one path with just that node. Otherwise, this algorithm returns paths obtained by appending the source node to all shortest paths from the nodes that are one step closer to the destination node, on each of the x and y axes.

Algorithm 1 Finding All Shortest Paths Between Two Nodes In Grids.
Input: The coordinates of two nodes: (x_1, y_1) and (x_2, y_2)
Output: Set of all shortest paths between them

$AllShortestPaths(x_1, y_1, x_2, y_2)$
1. $xStep = \begin{cases} 1 & \text{if } x_1 < x_2 \\ -1 & \text{if } x_1 > x_2 \\ 0 & \text{otherwise} \end{cases}$
2. $yStep = \begin{cases} 1 & \text{if } y_1 < y_2 \\ -1 & \text{if } y_1 > y_2 \\ 0 & \text{otherwise} \end{cases}$
3. **if** ($xStep == 0$ and $yStep == 0$) **RETURN** $\{\langle(x_1, y_1)\rangle\}$;
4. $S = \phi$;
5. **if** ($xStep \neq 0$)
$S = S \cup \{(x_1, y_1) :: P \mid P \in AllShortestPaths(x_1 + xStep, y_1, x_2, y_2)\}$;
6. **if** ($yStep \neq 0$)
$S = S \cup \{(x_1, y_1) :: P \mid P \in AllShortestPaths(x_1, y_1 + yStep, x_2, y_2)\}$;
7. **RETURN** S.

Time Complexity of Algorithm 1. Let $X = |x_1 - x_2|$, and $Y = |y_1 - y_2|$. There are $C^X_{(X+Y)}$ shortest paths between (x_1, y_1) and (x_2, y_2), where $C^X_{(X+Y)}$ is the number of ways of selecting X items from a set of $(X + Y)$ items. Finding each shortest path requires $O(X + Y)$ append operations. Therefore, the time complexity of Algorithm 1 is $O((X + Y)C^X_{(X+Y)})$.

Data Preservation With Load-balancing. Load-balancing data preservation algorithm (Algorithm 2) works as follows. First, we use minimum cost flow algorithm (Section 3.3) to achieve minimum energy consumption as well as to find the destination nodes of all data items (line 1). Then, for each data item, we find all the shortest paths between its SN and the destination node using Algorithm 1 (line 5). Finally, among all the shortest paths for this data item, we choose the one whose minimum energy-node has the maximum energy as its offloading path (line 7-11). This way it ensures that destination nodes with less energy do not relay data items, hence saving energy and preserving their stored data for longer time. Although we can not prove the optimality of this algorithm, we show in Section 5 that it performs better than the one without load-balancing.

Algorithm 2 Load-balancing Data Preservation Algorithm.
Input: The sensor network graph and set of data items D
Output: Set of offloading paths: $\{P_j | D_j \in D\}$
1. Run minimum cost flow algorithm;

2. **for** each source node with coordinate (x_1, y_1)
3. **for** each of its data item
4. Let the coordinate of its destination node be (x_2, y_2);
5. Get all the shortest paths from $AllShortestPaths(x_1, y_1, x_2, y_2)$;
6. $MaxMinEnergy = 0$;
7. **for** each such shortest path P_j
8. $MinEnergy[j]$ = minimum energy of nodes in P_j;
9. **if** $(MinEnergy[j] > MaxMinEnergy)$
$MaxMinEnergy = MinEnergy[j]$;
10. **end for;**
11. Find path, say P_k, with $MaxMinEnergy$, as its offloading path;
12. **end for;**
13. **end for;**
14. **RETURN** all the offloading paths.

Time Complexity of Algorithm 2. The length of any shortest path in a grid of $|V|$ nodes is at most $2\sqrt{|V|}$, since $\sqrt{|V|}$ is the number of nodes along either x or y axis. The largest number of shortest paths between any two nodes is therefore $C^{\sqrt{|V|}}_{2\sqrt{|V|}}$. The time complexity of the minimum cost flow algorithm is $O(|V|^2|E|log(|V|\mathcal{C}))$, which is less than $C^{\sqrt{|V|}}_{2\sqrt{|V|}}$. Therefore, the time complexity of Algorithm 2 is $O(q \times 2\sqrt{|V|} \times C^{\sqrt{|V|}}_{2\sqrt{|V|}})$.

5 Performance Evaluation

We first compare the network performance of the load-balancing data preservation algorithm with the minimum cost flow data preservation algorithm in [9], which does not employ load-balancing technique. We then compare energy consumption of data preservation using maximum flow algorithm and minimum cost flow algorithm. Since it is shown in Section 3.2 that Edmonds-Karp algorithm costs less energy than Ford-Fulkerson does, we adopt Edmonds-Karp as the maximum flow algorithm in the comparison.

Comparing load-balancing data preservation and minimum cost flow-based data preservation. The sensor network is a 10×10 grid network. We randomly choose two nodes in the network as SNs. The storage capacity of each destination node is 10, and the initial energy of each node (including the source nodes) is 1000. Fig. 4(a) shows that both algorithms cost the same amount of energy. However, Fig. 4(b) shows that with the increase of the number of data items at source nodes, the minimum remaining energy of the destination nodes given by load-balancing algorithm gets larger than that given by minimum cost flow algorithm. This indicates that data preservation would fail much later with load-balancing algorithm than without load-balancing.

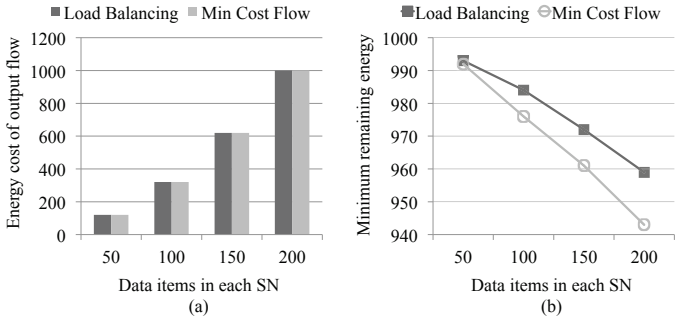


Fig. 4. Comparison between the load-balancing data preservation and the minimum cost flow-based data preservation.

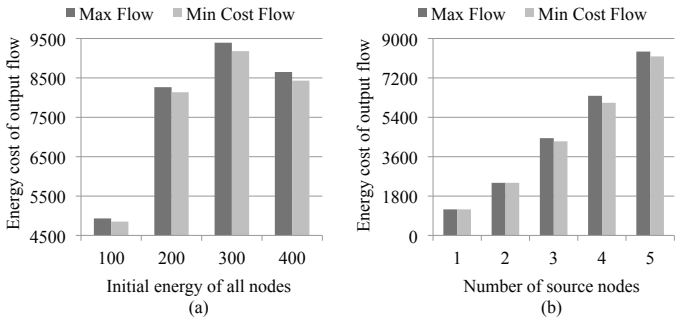


Fig. 5. Comparison between the Edmonds-Karp maximum flow algorithm and the minimum cost flow algorithm.

Comparing maximum flow and minimum cost flow. Since minimum cost flow algorithm gives minimum energy consumption for data preservation, and Edmonds-Karp costs much less energy than Ford-Fulkerson does, we ask the following question: *How close does Edmonds-Karp perform w.r.t. minimum cost flow?* Below we compare their performances. We set the network size as 15×15 and randomly choose 5 nodes as SNs, each with 400 data items. The storage capacity of each destination node is 10. Fig. 5(a) shows the energy consumption comparison by varying the initial energy level of each node. When initial energy equals 100, not all the data items can be offloaded. We thereby use Edmonds-Karp to first find the maximum amount of data items that can be offloaded by each SN, and use that information as input for minimum cost flow algorithm. It shows that the total energy consumption by Edmonds-Karp is larger than that of minimum cost flow. When initial energy is 200 and 300, all the data items can be offloaded from their SNs. When initial energy gets to 400, since each destination node has enough energy to either store or relay the data items, there are more shorter offloading paths available, therefore the total energy consumption decreases for both algorithms. In all cases, the total energy consumption

by Edmonds-Karp is larger than that of minimum cost flow. Fig. 5(b) uses the same parameters as in Fig. 5(a), except that now we fix the energy level of all the nodes as 200, and change the number of SNs. It shows that when there are only one or two SNs, Edmonds-Karp and minimum cost flow have similar performance. However, when the number of SNs increases, minimum cost flow costs less energy than Edmonds-Karp does. In all, minimum cost flow performs better than Edmonds-Karp in more stressed scenarios (i.e., more data items to offload).

6 Conclusion and Future Work

We study a generalized data preservation problem to preserve data inside sensor networks, considering that each node has limited battery power. We show that this problem can be modeled and solved as a minimum cost flow problem, which is solvable in polynomial time. By examining how different network flow algorithms can affect sensor network performance, we take a step further to view network flow problems from the perspective of efficient resource allocation, and study their applicability to sensor network scenarios. Our ongoing and future works are as follows. First, instead of a grid network, we will adopt a randomly generated sensor network for further study. Second, it would be interesting to prove if Edmonds-Karp costs the *minimum* amount of energy for data preservation, among all maximum flow algorithms. That is, when each edge has the same unit cost, is Edmonds-Karp a minimum cost maximum flow?[5] Third, we hope to study the problem using a more general energy model, wherein the energy consumption of sending data from one node to another depends on not only the size of the data but also the distance between nodes.

Acknowledgment

This work was supported in part by the NSF Grant CNS-1116849.

References

1. Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., Corke, P.: Data collection, storage, and retrieval with an underwater sensor network. In: Proc. of SenSys 2005. 154–165
2. Li, S., Liu, Y., Li, X.: Capacity of large scale wireless networks under gaussian channel model. In: Proc. of MOBICOM 2008. 140–151
3. Luo, L., Cao, Q., Huang, C., Wang, L., Abdelzaher, T., Stankovic, J.: Design, implementation, and evaluation of enviromic: A storage-centric audio sensor network. ACM Transactions on Sensor Networks **5**(3) (2009) 1–35
4. Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., Welsh, M.: Fidelity and yield in a volcano monitoring sensor network. In: Proc. of OSDI 2006. 381–396

[5] Note that minimum cost maximum flow problem is to find a maximum flow that has the minimum cost among all the maximum flows, considering that each edge has both a capacity and a cost.

5. Martinez, K., Ong, R., Hart, J.: Glacsweb: a sensor network for hostile environments. In: Proc. of SECON 2004. 81–87
6. Jain, S., Shah, R., Brunette, W., Borriello, G., Roy, S.: Exploiting mobility for energy efficient data collection in wireless sensor networks. MONET **11**(3) (2006) 327–339
7. Jea, D., Somasundara, A.A., Srivastava, M.B.: Multiple controlled mobile elements (data mules) for data collection in sensor networks. In: Proc. of the IEEE DCOSS. (2005) 244–257
8. Mathioudakis, I., White, N.M., Harris, N.R.: Wireless sensor networks: Applications utilizing satellite links. In: Proc. of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007). (2007) 1–5
9. Tang, B., Jaggi, N., Wu, H., Kurkal, R.: Energy efficient data redistribution in sensor networks. ACM Transactions on Sensor Networks **9**(2) (May 2013) 1–28
10. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
11. Hou, X., Sumpter, Z., Burson, L., Xue, X., Tang, B.: Maximizing data preservation in intermittently connected sensor networks. In: Proc. of IEEE MASS 2012. 448–452
12. Maulin Patel, S. Venkatesan, R.C.: Energy efficient capacity constrained routing in wireless sensor networks. International Journal of Pervasive Computing and Communications **2** (2006) 69–80
13. Bodlaender, H.L., Tan, R.B., Dijk, T.C., Leeuwen, J.: Integer maximum flow in wireless sensor networks with energy constraint. In: Proc. of the 11th Scandinavian workshop on Algorithm Theory, SWAT 08. 102–113
14. Hong, B., Prasanna, V.K.: Maximum data gathering in networked sensor systems. Intl J. Distributed Sensor Networks **1** (2005) 57–80
15. Xue, Y., Cui, Y., Nahrstedt, K.: Maximizing lifetime for data aggregation in wireless sensor networks. Mob. Netw. Appl. **10**(6) (December 2005) 853–864
16. Ghiasi, S., Srivastava, A., Yang, X., Sarrafzadeh: Optimal energy aware clustering in sensor networks. Sensors **2**(7) (2002) 258–269
17. Xue, X., Hou, X., Tang, B., Bagai, R.: Data preservation in intermittently connected sensor networks with data priorities. In: Proc. of IEEE SECON 2013. 65–73
18. Ha, R.W., Ho, P.H., Shen, X.S., Zhang, J.: Sleep scheduling for wireless sensor networks via network flow model. Comput. Commun. **29** (August) 2469–2481
19. Papadimitriou, C., Yannakakis, M.: Optimization, approximation and complexity classes. Journal of Computer and System Sciences **43** (1991) 425 – 440
20. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proc. of HICSS 2000
21. Goldberg, A.V.: An efficient implementation of a scaling minimum-cost flow algorithm. Journal of Algorithms **22**(1) (1997) 1–29
22. Goldberg, A.V.: Andrew Goldberg's network optimization library http://www.avglab.com/andrew/soft.html.
23. Hershberger, J., Maxel, M., Suri, S.: Finding the k shortest simple paths: A new algorithm and its implementation. ACM Trans. Algorithms **3**(4) (2007) 1–19
24. Eppstein, D.: Finding the k shortest paths. SIAM J. Computing **28**(2) (1998) 652–673