Integer Maximum Flow in Wireless Sensor Networks with Energy Constraint

Hans L. Bodlaender

Richard B. Tan

Thomas C. van Dijk

Jan van Leeuwen

Technical Report UU-CS-2008-005
Februari 2008

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Integer Maximum Flow in Wireless Sensor Networks with Energy Constraint ⋆

Hans L. Bodlaender [a,1] Richard B. Tan [a,b]
Thomas C. van Dijk [a] and Jan van Leeuwen [a]

[a] *Department of Information and Computing Sciences*
Utrecht University
Utrecht, The Netherlands
{hansb,rbtan,thomasd,jan}@cs.uu.nl

[b] *Department of Computer Science*
University of Sciences & Arts of Oklahoma
Chickasha, OK, U.S.A.

Abstract

We study the integer maximum flow problem on wireless sensor networks with energy constraint. In this problem, sensor nodes gather data and then relay them to a base station, before they run out of battery power. Packets are considered as integral units and not splittable. The problem is to find the maximum data flow in the sensor network subject to the energy constraint of the sensors. We show that this *integral* version of the problem is *strongly* NP-complete and in fact APX-hard. It follows that the problem is unlikely to have a polynomial time approximation scheme. Even when restricted to graphs with concrete geometrically defined connectivity and transmission costs, the problem is still strongly NP-complete. We provide some interesting polynomial time algorithms that give good approximations for the general case nonetheless. For networks with bounded treewidth greater than two, we show that the problem is *weakly* NP-complete and provide pseudo-polynomial time algorithms. For a special case of graphs with treewidth two, we give a polynomial time algorithm.

⋆ This work was supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

[1] Corresponding author.

1 Introduction

A wireless sensor (or smart dust) is a small physical device that contains a microchip with a miniature battery, and a transmit/receive capability of limited range. Sensors have been deployed in different environments to gather data, perform surveillances and monitor situations in diverse areas such as military, medical, traffic and natural environments. (See e.g. Zhao and Guibas [21].)

As the battery power of a sensor is limited and non-replaceable, it is crucial to maximize the lifetime of the wireless sensor network to ensure the continuing function of the whole network. We study the situation of a data-gathering sensor network, where sensors are deployed in the field to gather data and then relay the data packets via other sensors back to a base station. It is desirable to get as many data packets as possible from the source sensors to the base station, before some of the sensor batteries are depleted. This then becomes an instance of the maximum flow problem, subject to the energy constraint of the lasting battery power of each sensor.

Most of the research papers for the maximum flow problem with energy constraint on wireless sensor networks (e.g. [8,9,12,16–18,20]) cast the problem into a Linear Programming (LP) form and assume *fractional* flows, i.e., splitting of packets into fractional portions is allowed. The corresponding LP-formulations then have polynomial time algorithms. These papers then present several heuristics that speed up the algorithms and compare various simulation results. A few papers [8,12,16,17] modified the Polynomial Time Approximation Scheme (PTAS) of Garg and Könemann [14] to obtain fast approximation algorithms.

As data packets are usually quite small, there are situations where splitting of packets into fractional ones is not desirable nor practical. We consider a model where data packets are considered as units that cannot be split, i.e. the packet flows are of *integral* values only. We call this the *Integer* Maximum Flow problem for Wireless Sensor Network with Energy Constraint: the Integer Max-Flow WSNC problem. The corresponding LP formulation becomes Integer Programming (IP) and may no longer have a polynomial time solution.

We show that the problem is in fact *strongly* NP-complete, and thus unlikely to have a Fully Polynomial Time Approximation Scheme (FPTAS) or a pseudo-polynomial time algorithm, unless P=NP. This result also holds for a class of graphs with geometrically defined connectivity and transmission costs, even when the nodes lie on a line. Furthermore, we show that even for a special fixed range model, the problem is APX-hard, thus unlikely to have even a PTAS (unless P=NP). We also provide some approximation algorithms for

the problem that do give good approximations nonetheless.

Many hard problems have polynomial time solutions when restricted to networks with bounded treewidth (see e.g. [4]). However, we show that for networks with bounded treewidth greater than two, the Integer Max-Flow WSNC problem is *weakly* NP-complete. We provide pseudo-polynomial time algorithms to compute integer maximum flows in this case. For a special case of graphs that have treewidth two, namely those graphs that have treewidth two when we add an edge from the single source to the sink, we provide a polynomial time algorithm.

The paper is organized as follows. In Section 2, we describe the model and the problem in detail. Section 3 covers the complexity issues. We show here the various NP-completeness results and describe some approximation algorithms. Section 4 contains the results for networks with bounded treewidth.

2 Preliminaries

In this section, we discuss the model we use and the precise formulation of the Integer Max-Flow WSNC problem. We also discuss some variants of the problem, and give some definitions used in other sections.

2.1 The Model

Our model of a sensor is based on the *first order radio model* of Heinzelman et al. [15]. A sensor node has limited battery power that is not replenishable. It consumes an amount of energy $\epsilon_{elec} = 50nJ/bit$ to run the receiving and transmitting circuitry and $\epsilon_{amp} = 100pJ/bit/m^2$ for the transmitter amplifier. In order to receive a k-bit data packet, a sensor has to expend $\epsilon_{elec} \cdot k$ energy, while to transmit the same packet from sensor i to sensor j will cost $\epsilon_{elec} \cdot k + \epsilon_{amp} \cdot k \cdot d_{ij}^2$ energy, where d_{ij} is the distance between sensors i and j.

We model a wireless sensor network as a *directed* graph $G = (N, A)$, where $N = \{1, 2, \ldots, n\} \cup \{t\}$ are the n sensor nodes along with a special non-sensor sink node t, and A is the set of directed arcs ij connecting node i to node j, $i, j \in N$. A sensor node i has energy capacity E_i and each arc ij has cost e_{ij}, the energy cost of receiving (possibly from some node) a packet and then transmitting it from node i to node j. We assume that no data is held back in intermediate nodes $\neq t$, i.e., data that flows in will flow out again, subject to the battery constraint of these nodes. All E_i's and e_{ij}'s are non-negative *integer* values.

The general model assumes that each sensor can adjust its power range for each transmission. We also consider in the next section the fixed range model, where each sensor has only a few *fixed* power settings. All our graphs are assumed to be *connected*. For each arc $ij \in A$, there is a directed path from a source node to the sink node that uses this arc.

2.2 The Problem

Given a wireless sensor network G, there is a set S of *source* sensor nodes, used for gathering data. The sink node t is a *base station* and is equipped with electricity and thus has unlimited energy to receive all packets. The remaining nodes are just *relaying* nodes, used to transfer data packets from the source nodes to the sink node. One would like to transmit as many packets as possible from the source nodes to the sink node. This is feasible as long as the battery power in the network suffices to do so. The transmission process can be viewed as a *flow* of packets from the sources to the sink. The problem is then to find the *maximum flow* of data packets in the network subject to the battery power constraint.

We assume that the data packets are quite small, thus it is neither reasonable nor practical to split them further into fractional portions. A *flow* f_{ij} is a function that assigns to each arc ij a non-negative integer value. This corresponds to the number of packets being sent via the arc ij. A flow is a *feasible* flow if $\sum_j f_{ij} \cdot e_{ij} \leq E_i$ for all nodes $i \in N$, where the sum is taken over all j with $ij \in A$; i.e., the flow through a node cannot exceed the battery capacity of the node.

We can now formulate the maximum flow problem for wireless sensor networks as the problem of determining the maximum number of packets that can be received by the sink node. We call this problem the Integer Maximum Flow problem for Wireless Sensor Networks with energy Constraints or *Integer Max-Flow WSNC problem* for short. The problem has the following Integer Linear Programming formulation.

The Integer Max-Flow WSNC problem:
Objective: maximize $F = \sum_{j \in N} f_{jt}$, t is the sink node,
subject to the following constraints:

$$f_{ij} \ \ integer, \qquad \forall ij \in A \qquad (1)$$

$$f_{ij} \geq 0, \qquad \forall ij \in A \qquad (2)$$

$$\sum_{j \in N} f_{ij} = \sum_{j \in N} f_{ji}, \qquad \forall i \in N - S - \{t\} \qquad (3)$$

$$\sum_{j \in N} f_{ij} \cdot e_{ij} \leq E_i, \qquad \forall i \in N \qquad (4)$$

Condition (3) is the conservation of flow constraint. It simply states that with the exception of the source and sink nodes, every node must send along the packets that it has received. Condition (4) is the energy constraint for the feasible flow: the energy needed to (receive and) transmit packets must be within the capacity of the battery power of each node. This condition also distinguishes the (integer) max-flow WSNC problem from the standard max-flow problem: there, the constraint condition is just $f_{ij} \leq c_{ij}$, where c_{ij} is the flow capacity of arc ij.

We note that without loss of generality, we can augment the network with a *super source* node s with unlimited energy to send and connect it to all the source nodes with some fixed cost. We can then view the network as having a single source and a single sink with a single commodity, subject to the battery energy constraint. However, note this may affect the treewidth of the network; the results for networks of bounded treewidth in section 4.1 assume a single source.

2.3 Other Variants

Other variants of the problem formulation exist for wireless sensor networks with energy constraint. For example, Floréen et al. [12] use the following energy constraint in their LP-formulation of the problem:

$$\sum_{j \in N} \tau_{ij} \cdot f_{ij} + \sum_{j \in N} \rho \cdot f_{ij} \leq E_i, \ \forall i \in N,$$

where the parameters τ_{ij} is the energy expended in sending a packet from node i to node j and ρ is the corresponding energy for receiving a packet. Their *objective* function in the LP-formulation is also slightly different and has some extra parameters that are non-integral.

Chang and Tassiulas [9] give an LP-formulation similar to ours for the single commodity case, except they formulate the problem over the set of all

paths from s to t and allow fractional packets. They also consider the multi-commodity case.

2.4 Definition of Treewidth

We define the concept of treewidth for a general (undirected) graph G^u. This concept is used in Section 4.

Definition 1 *A tree-decomposition of a graph $G^u = (N, A^u)$ is a pair $D = (X, T)$ with $T = (P, F)$ a tree and $X = \{X_p \mid p \in P\}$ a family of subsets of N, one for each node of T, such that*

- $\bigcup_{p \in P} X_p = N$,
- *for every edge $\{i, j\} \in A^u$, there exists a $p \in P$ with $i \in X_p$ and $j \in X_p$,*
- *for all $p, q, r \in P$: if q is on the path from p to r in T, then $X_p \cap X_r \subseteq X_q$.*

The treewidth of a tree-decomposition $(\{X_p \mid p \in P\}, T = (P, F))$ is $\max_{p \in P} |X_p| - 1$. The treewidth of a graph G^u is the minimum treewidth over all possible tree-decompositions of G^u.

An undirected graph $G^u = (N, A^u)$ is said to be a *minor* of a graph $H^u = (M, B^u)$, if G^u can be obtained from H^u by a series of vertex deletions, edge deletions, and edge contractions; where an edge contraction is the operation that takes two adjacent vertices i and j, and replaces it by a new vertex, adjacent to all vertices that were adjacent to i or j. It is well known that the treewidth does not increase when taking minors.

In this paper, the treewidth of a directed graph G is just the treewidth of the underlying undirected graph G^u, i.e., the graph obtained by dropping direction of edges.

3 Complexity

We will first look at the complexity of the problem on general graphs with arbitrary costs at the arcs. We show the problem is NP-complete. Next we show this proof carries over to a restriction of the problem to a class of graphs with geometrically defined connectivity and energy consumption. Then we look at another restriction of the problem, on general graphs again, but with only a fixed amount of distinct energy costs. The problem is polynomial time solvable for one energy level, but with more distinct power levels the problem is shown to be APX-hard. Lastly, we demonstrate a polynomial time approximation

algorithm for the general case. Note however that this algorithm does not guarantee a constant approximation ratio.

3.1 General graphs

Since each data packet is a self-contained unit and cannot be split, the corresponding LP formulation is an Integer Programming (IP) formulation and may no longer have a polynomial time solution. In fact, we prove that the problem is *strongly* NP-complete.

Theorem 2 *The decision variant of the Integer Max-Flow WSNC problem is* strongly *NP-complete.*

PROOF. We reduce the 3-Partition problem to the decision version of the Integer Max-Flow WSNC problem.

> 3-Partition
> Instance: Given a multiset $\mathcal{S}$ of $n = 3m$ positive integers, where each $x_i \in \mathcal{S}$ is of size $B/4 < x_i < B/2$, for a positive integer B.
> Question: Can $\mathcal{S}$ be partitioned into m subsets (each necessarily containing exactly three elements) such that the sum of each subset is equal to B?

The 3-Partition Problem is strongly NP-complete [13].

For any instance I of the 3-Partition problem we create an instance I' of a wireless sensor network as follows. Each number $x_i \in \mathcal{S}$ corresponds to a sensor relay node r_i. Additionally, we have m source nodes $s_1, \ldots, s_m$ each having exactly B energy. The source nodes play the role of the subsets. Now connect each of the source nodes s_j with all the relay nodes r_i with arc cost $e_{ji} = x_i$. The intention is that it will cost each source node exactly x_i energy to send one packet to relay node r_i. We further connect all the relay nodes to a sink node t. Each relay node r_i has energy $E_i = B$ and arc cost $e_{it} = B$, just sufficient energy to send only one packet to the sink node.

Then our instance of the 3-Partition problem has a partition into m subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m$, each with sum equals B if and only if for each subset $\mathcal{S}_i = \{x_{i1}, x_{i2}, x_{i3}\}$ the source node s_i sends three packets, one each to relay nodes r_{i1}, r_{i2}, r_{i3} consuming the energies x_{i1}, x_{i2}, x_{i3}, thus draining all of its battery power of $B = \sum_{j=1}^{3} x_{ij}$. This will give a maximum flow of $n = 3m$ packets for the whole network. Thus, 3-Partition reduces to the question whether the WSNC network can transmit at least $3m$ packets to the sink.

We have now given a pseudopolynomial reduction (see [13]), and thus we have shown that the Integer Max-Flow WSNC problem is strongly NP-complete. □

Corollary 3 *The Integer Max-Flow WSNC problem has no fully polynomial time approximation scheme (FPTAS) and no pseudo-polynomial time algorithm, unless P=NP.*

PROOF. This follows from the fact that a strongly NP-complete problem has no FPTAS and no pseudo-polynomial time algorithm, unless P=NP. (See [13]). □

We show later on that even in a restricted case, the problem is APX-hard, i.e. the problem does not even have a PTAS unless P=NP.

3.2 The Geometric Model

We will now look at the geometric version of the problem in which the nodes are concretely embedded in space. In this version, each node has a position, and transmitting to a node at distance d costs d^2 energy. This quadratic cost is a typical model of radio transmitters.

Definition 4 *A* geometric configuration *is a complete graph where each node $i \in N$ has a location $p(i)$ and initial battery capacity E_i. The cost of the edge between vertices i and j is $e_{ij} = |p(i) - p(j)|^2$.*

In this section we show that the Integer Max-Flow WSNC-problem remains NP-complete when restricted to geometric configurations where all nodes lie on a line. We first consider the case where we allow variable battery capacities E_i (Theorem 5) and then give a more elaborate construction, still with all nodes on a line, where all nodes have equal battery capacity (Theorem 10).

Theorem 5 *The Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations on the real line.*

PROOF. Again, we give a pseudopolynomial reduction from 3-Partition. First we describe how to construct a geometric configuration $\mathbf{C}$ on the real line, where the Integer Max-Flow WSNC-problem is equivalent to a given instance of 3-Partition. We then construct an equivalent configuration $\mathbf{C_P}$ that can be described in polynomial size. These steps together show that the Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations.

Like before, we have m source nodes $s_1, \ldots, s_m$. Each starts with $B = \sum x_i/m$ energy. These source nodes again play the role of the subsets and we place

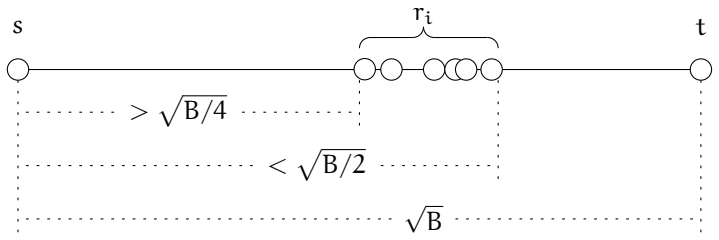

Fig. 1. Configuration **C**.

them all at the origin of our geometric configuration, i.e., $p(s_i) = 0$ for all i. (This also means that the cost of sending from one source node to another is zero, but such flow can be disregarded since all these nodes are sources.)

Corresponding to each $x_i \in \mathcal{S}$ we have a 'relay' node r_i that serves the same purpose as before: to receive one packet from a source node, costing x_i energy for this source node, and relay the packet to the sink. By setting $p(r_i) = \sqrt{x_i}$ we achieve that a source node must use x_i energy to send a message to r_i.

Finally, we place a sink node t with $p(t) = \sqrt{B}$. We want each relay node to have just enough energy to send exactly one packet to the sink, so we set $E_{r_i} = (p(t) - p(r_i))^2 = B + x - 2\sqrt{B}\sqrt{x}$.

This concludes the construction of our geometric configuration **C**. The construction is illustrated in Figure 1.

Lemma 6 *Suppose we have a flow of value n in* **C**. *Then every relay node receives exactly one packet from a source and sends it to the sink.*

PROOF. If n packets reach the sink, then n packets must have left the sources. By the restriction on the values x_i of the 3-Partition instance, each edge leaving the sources costs strictly more than $B/4$ energy. Since the source nodes start with B energy, no source node can send more than 3 packets. There are only $m = n/3$ source nodes, so every source node must send exactly 3 packets. In particular, no packets are sent directly from a source node to the sink, as this would use up all energy of the source node. Therefore, the sink node only receives packets from relay nodes. No relay node can afford to send more than one packet to the sink, so in fact every relay node sends exactly one packet to the sink. □

Using Lemma 6, the following can now be shown in the same way as Theorem 2 for the case on arbitrary graphs.

Proposition 7 *The configuration* **C** *has a solution of the Integer Max-Flow WSNC-problem with n packets if and only if the corresponding 3-Partition instance is* **Yes**.

Note that this does not yet give an NP-hardness proof for the Integer Max-Flow WSNC-problem on geometric configurations on the line, as configuration $\mathbf{C}$ has nodes at real-valued coordinates: the specification of the location of the points in $\mathbf{C}$ contains square roots. We shall now construct a geometric configuration $\mathbf{C_P}$ which is equivalent to $\mathbf{C}$, but whose positions are all polynomially representable rational numbers.

We do this by choosing the locations as integer multiples of some ε (value to be determined later), rounding down. The initial power of the batteries also needs to be quantized. We give the source nodes exactly B energy, which is already integer. We give the relay nodes precisely enough energy to send one packet to the sink; this amount can be calculated from the actual distance in $\mathbf{C_P}$.

Lemma 8 *The value for ε can be chosen such that $\mathbf{C_P}$ is equivalent to $\mathbf{C}$ and can be represented in polynomial size.*

PROOF. Consider a grid of precision ε, onto which the positions are rounded down. Rounding down makes communication from sources to relay nodes cheaper. For the NP-completeness construction we need that a source cannot send 4 packets. From the 3-Partition instance we have that the x_i are strictly bigger than $B/4$. Since the x_i are integer, we have in particular that the $x_i \geq B/4 + \frac{1}{4}$. This gives the following constraint on ε:

$$4\left(\sqrt{\frac{B+1}{4}} - \varepsilon\right)^2 \;>\; B.$$

This is equivalent to

$$\varepsilon \;<\; \frac{\sqrt{B+1} - \sqrt{B}}{2}. \tag{1}$$

That is, if we round coordinates down to a multiple of ε and (1) holds, no source node can send more than three packets. Note that this bound is $\Theta(1/\sqrt{B})$.

There is a further constraint on ε. We do not want to enable flows that do not correspond to a 3-Partition. Therefore, a source node should, with its B energy, not be able to send packets to a set of relay nodes if the corresponding x_i sum to strictly more than B. This gives the following constraint on ε, for all $a, b, c \in \mathcal{S}$:

$$a + b + c \geq B + 1 \implies (\sqrt{a} - \varepsilon)^2 + (\sqrt{b} - \varepsilon)^2 + (\sqrt{c} - \varepsilon)^2 > B.$$

This holds when the following condition holds; we solve the equation by setting $a + b + c = B + 1$.

$$\varepsilon \;<\; \frac{\alpha - \sqrt{\alpha^2 - 3}}{3}, \quad \textbf{where} \;\; \alpha = \sqrt{a} + \sqrt{b} + \sqrt{c}.$$

This upperbound for ε is monotonically decreasing in α (for $\alpha > \sqrt{3}$) and is therefore minimized by maximizing α. This occurs at $a = b = c = (B+1)/3$, giving

$$\varepsilon \;<\; \frac{\sqrt{B+1} - \sqrt{B}}{\sqrt{3}}. \tag{2}$$

This bound on ε is also $\Theta(1/\sqrt{B})$.

The number of gridpoints for a grid of length $\sqrt{B}$ is then

$$\frac{\sqrt{B}}{\varepsilon} \;=\; \frac{\sqrt{B}}{\Theta(1/\sqrt{B})} \;=\; \Theta(B).$$

A position can therefore be represented in $\Theta(\log B)$ space, just like the numbers in the given 3-Partition instance (which are between $B/4$ and $B/2$).

Finally, we must specify ε. We can take e.g.

$$\varepsilon = \frac{1}{5\lceil\sqrt{B}\rceil}.$$

This satisfies constraints (1) and (2) and can be written as a rational number in polynomial space and time. Note that we can also compute the coordinates of all relay nodes and the sink in polynomial time. □

The proof of Theorem 5 now follows from Lemmata 6, 8 and Proposition 7: the Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations, even when restricted to a line. □

The nodes in $\mathbf{C_P}$ have non-integer positions, but the following shows that this is not essential.

Corollary 9 *The Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations on a line, where each node has an integer coordinate.*

PROOF. Transform $\mathbf{C_P}$ as follows. Multiply each position by ε^{-1} and each battery capacity by ε^{-2}. This transformation assures that all coordinates are integer, since all positions in $\mathbf{C_P}$ are multiples of ε. Also, the transformed geometric configuration is equivalent to $\mathbf{C_P}$, since for any set of distances d_i, a bound E and a value for ε, we have that the old situation is equivalent to the transformed situation:

$$\sum {d_i}^2 \leq E \iff \sum \left(\frac{d_i}{\varepsilon}\right)^2 \leq \frac{E}{\varepsilon^2}. \qquad \square$$

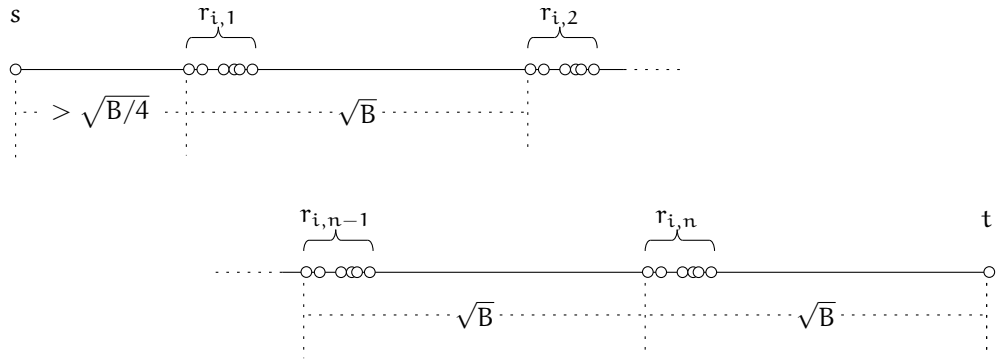

Fig. 2. Configuration $\mathbf{C}'$.

We finish this section by proving that Theorem 5 still holds when all battery capacities are equal.

Theorem 10 *The Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations on the real line, even when all battery capacities are equal.*

PROOF. Recall configuration $\mathbf{C}$: source nodes, a cluster of relay nodes and a sink node. The battery capacity of the relay nodes was used to coerce each relay node to send exactly one packet, and to send it to the sink. Doing this for all relay nodes at once required varying battery capacities; we shall now use a more elaborate construction $\mathbf{C}'$.

Where before we had a single cluster of relay nodes, we shall now have n clusters of relay nodes, spaced $\sqrt{B}$ apart. Then the geometric configuration $\mathbf{C}'$ is as follows.

- Source nodes: $s_1 \dots s_m$, with $p(s_i) = 0$.
- Relay nodes: for each x_i, there are n relay nodes, $r_{i,1} \dots r_{i,n}$, with $p(r_{i,j}) = \sqrt{x_i} + (j-1)\sqrt{B}$.
- A sink node t, with $p(t) = n\sqrt{B} + \sqrt{x_1}$.
- All nodes have starting battery capacity of B.

The configuration is visualized in Figure 2. We shall again prove that a flow of value n is possible in this configuration if and only if there is a 3-partition in the given instance.

For convenience, let the x_i be indexed in order of non-decreasing value. Furthermore, consider multiple relay nodes at the exact same location, which occurs if $x_i = x_j$ for some $i \neq j$. Then the relay nodes for x_i and x_j are interchangeable in any solution. This allows us to proceed in the following proof as if $i < j$ implies $x_i < x_j$.

First, suppose there is a solution to the given instance of 3-Partition. Similar to the earlier proofs, each source node in the configuration can send three packets, such that each relay node in the first cluster (i.e., each $r_{i,1}$) receives exactly one packet. Now, each relay node except those in the last cluster forwards its packet to $r_{i,j+1}$; the relay nodes in the last cluster (i.e., all $r_{i,n}$) forward the packet to the sink. This gives a flow of value n.

Now suppose we have a flow of value n in $\mathbf{C}'$. We will show that the given instance of 3-Partition has a solution. Note that the gap between the last relay node in the last cluster and the sink is

$$\begin{aligned} p(t) - p(r_{n,n}) &= \sqrt{B} + \sqrt{x_1} - \sqrt{x_n} \\ &> \sqrt{B} + \sqrt{B/4} - \sqrt{B/2} \approx 0.79\sqrt{B}. \end{aligned}$$

The energy required for sending more than one packet over this distance —i.e., from any node in the last cluster to the sink— is at least twice this distance squared ($\approx 1.26B$), which no node can afford. So if there is n flow in the network, each relay node in the last cluster must send exactly one packet to the sink. Similarly, every node in cluster $j < n$ must send exactly one packet to a node in cluster $j + 1$.

We cannot rule out that some relay nodes send packets to other nodes in the same cluster, in addition to their packet to the next cluster. We shall prove that some nodes (and in particular, all nodes in the first cluster) cannot do this.

First consider $r_{1,1}$, the leftmost relay node. By the preceding argument, it must send a packet to the next cluster. In particular, this must be $r_{1,2}$: all other nodes in cluster 2 are more than $\sqrt{B}$ away and therefore cannot be reached. Note that sending this particular packet depletes the battery of $r_{1,1}$ exactly.

Definition 11 *A node is called* forced *if, when there is n flow in the network, it necessarily spends all its energy to send exactly one packet.*

As we have just argued, $r_{1,1}$ is *forced*. The same argument shows that $r_{1,2}$ must send the packet to $r_{1,3}$ and so forth, on to the sink. This way, all $r_{1,j}$ are forced. Therefore, if there is a flow of n packets, one of the packets must take the path $p_1 = (\ s_i,\ r_{1,1},\ r_{1,2},\ \ldots,\ r_{1,n},\ t\ )$, for some source node s_i.

Lemma 12 *For all i, j, if $i + j \leq n + 1$ then relay node $r_{i,j}$ is forced.*

PROOF. We just argued that all $r_{1,j}$ are forced. We now look at $r_{2,1}$, the second relay node in the first cluster. It must send a packet to the next cluster. It cannot send it to $r_{1,2}$, however: the battery of that node is already completely depleted by the packet it must send for path p_1, so any packets sent there would

not be able leave. The packet can be sent to $r_{2,2}$, at cost exactly B, but it cannot be sent any further. So $r_{2,1}$ is also forced, and again there is only one choice for where to send the packet: via all the relay nodes for x_2. This way, all $r_{2,j}$ are forced, except for $r_{2,n}$: the distance between $r_{2,n}$ and t is less than $\sqrt{B}$ and its battery is not exhausted at once.

We now show the lemma with induction on i. We argued above the cases $i = 1$ and $i = 2$. Suppose the result holds for $i-1$, $i \geq 3$. Consider a relay node $r_{i,j}$, $i+j \leq n+1$. This node $r_{i,j}$ must send a packet to cluster $j+1$. Call the node it sends the packet to $r_{i^*,j+1}$. Clearly, $i^* \leq i$, since otherwise the transmission would cost too much energy. We obtain a contradiction if $i^* < i$. In that case $r_{i^*,j}$ is forced (induction hypothesis), so it sends a packet to $r_{i^*,j+1}$. Also, $r_{i^*,j+1}$ is forced (again, induction hypothesis), so it sends exactly one packet. Thus $r_{i,j}$ cannot send a packet to $r_{i^*,j+1}$. This leaves $i^* = i$ as the only option, and hence $r_{i,j}$ is forced. □

In particular, we have that all $r_{i,1}$ are forced. This means that all relay nodes in the first cluster receive exactly one packet from a source node, and spend all their energy sending this single packet to the next cluster. This is only possible if the original x_i have a valid 3-partition: there is n flow in $\mathbf{C}'$ if and only if the original 3-Partition instance has a solution.

Again, we need to consider polynomial representation. We will round down the positions in $\mathbf{C}'$ to multiples of some ε and call the resulting geometric configuration $\mathbf{C}'_{\mathbf{P}}$.

Lemma 13 *The value for ε can be chosen such that $\mathbf{C}'_{\mathbf{P}}$ is equivalent to $\mathbf{C}'$ and can be represented in polynomial size.*

PROOF. In addition to constraints (1) and (2) there is now an additional constraint. The construction requires that the *forced* nodes expend all their energy reaching a node at distance $\sqrt{B}$. After rounding, this distance might be only $\sqrt{B} - \varepsilon$. This is no problem as long as the node's remaining energy is not enough to reach any other node. The smallest distance between two nodes, again considering the rounding, is $\sqrt{B/2} - \sqrt{(B-1)/2} - \varepsilon$. This gives the constraint

$$(\sqrt{B} - \varepsilon)^2 \ + \ (\sqrt{B/2} - \sqrt{(B-1)/2} - \varepsilon)^2 \ > \ B \tag{3}$$

This bound on ε is $\Theta(\frac{1}{B\sqrt{B}})$ and can be satisfied by choosing e.g.

$$\varepsilon = \frac{1}{5B\lceil\sqrt{B}\rceil}.$$

This yields a total of $\Theta(nB^2)$ grid points, leading to $\Theta(\log nB)$ space per node. □

This concludes the proof of Theorem 10: the Integer Max-Flow WSNC-problem is strongly NP-complete on geometric configurations on a line, even when all nodes have equal battery capacity. □

3.3 Fixed Range Model

Now we return to the case for arbitrary graphs. Suppose that every sensor node has only a *fixed* number of power settings. For example, there may be only *one* setting, so that every node within the range is considered a neighbor; or perhaps there are only *two* settings: *short* and *long* range power settings.

It turns out that for the case when there is only one power setting, there is an easy solution. Since now the energy cost e_{ij} is the same for all neighbors j, the maximum flow capacity $f_{ij} = \lfloor E_i/e_{ij} \rfloor$ is also fixed for all outgoing arcs of node i. We can then transform the sensor network into a regular flow network by using the splitting technique in flow networks as follows. (See e.g. the book by Ahuja, et al. [1].) Split each node i into two nodes i and i' and connect them with an arc with capacity $c_{ij} = \lfloor E_i/e_{ij} \rfloor$. The capacity of the original arcs ij will also all be set to c_{ij}, for all $ij \in A$. We then have a new graph that is a flow network with twice as many nodes and n additional arcs. Then it is easy to see that this variant of the Max-Flow WSNC problem is just the standard Max-Flow Min-Cut problem and has a polynomial time algorithm of $O(n^3)$, even in the integer case. This fact has also been noted by Chang and Tassiulas [8]. For the sake of completeness, we record this fact below.

Theorem 14 *If there is only one power setting at each sensor node, then there is a polynomial time algorithm to solve the Integer Max-Flow WSNC problem.*

The situation changes when the number of fixed power settings is increased to two.

Theorem 15 *If there are two power settings at each sensor node, then there is no PTAS for the Integer Max-Flow WSNC problem, unless P=NP.*

PROOF. We reduce a restricted version of the Generalized Assignment Problem (GAP) by Chekuri and Khanna [10] to the Integer Max-Flow WSNC problem with two power settings.

2-SIZE 3-CAPACITY GENERALIZED ASSIGNMENT PROBLEM (2GAP-3)
INSTANCE: A set $\mathcal{B}$ of m bins and a set $\mathcal{S}$ of n items. Each bin j has capacity $c(j) = 3$ and for each item $i \in \mathcal{S}$ and bin $j \in \mathcal{B}$, we are given a size $s(i,j) = 1$ or $s(i,j) = 1 + \delta$ (for some $\delta > 0$) and a profit $p(i,j) = 1$.
OBJECTIVE: Find a subset $\mathcal{U} \subseteq \mathcal{S}$ of maximum profit such that $\mathcal{U}$ has a

feasible packing in $\mathcal{B}$.

Chekuri and Khanna [10] show that the 2GAP-3 problem is APX-hard, hence it does not have a PTAS (unless P=NP).

Given an instance I of 2GAP-3 we create an instance I' of the Integer Max-Flow WSNC as follows. For each bin $j \in \mathcal{B}$ we have a source node j' with energy capacity $E_{j'} = 3$. Corresponding to each item $i \in \mathcal{S}$ we have a relay node i'. Each source node j' is connected to each of the relay nodes i' by an arc $j'i'$ with energy cost $e_{j'i'} = 1$ if $s(i,j) = 1$ and $e_{j'i'} = 1+\delta$ if $s(i,j) = 1+\delta$. We also have one sink node t. Each of the sensor relay node i' is further connected to the sink node t and provided with just sufficient battery power to have the arc energy cost to send only one packet to the sink node, i.e., we set $E_{i'} = 1$ and $e_{i't} = 1$.

As each node i' that represents an item can forward only one packet, each arc of the form $j'i'$, $j \in \mathcal{B}$, $i \in \mathcal{S}$ can also carry at most one packet. Thus, there is a one-to-one correspondence between integer flows in I fulfilling energy constraints, and feasible packings of sets of items $\mathcal{U} \subseteq \mathcal{S}$: an item i that is placed in bin j corresponds to a unit of flow that is transmitted from j' to i' and then from i' to t. The value of the flow equals the total profit of the packed items.

Thus, we can observe that our reduction is an AP-reduction. As AP-reductions preserve APX-hardness (see e.g., [3]), we can conclude the theorem. □

3.4 Approximation Algorithms

As the Integer Max-Flow WSNC problem has no PTAS (unless P=NP), our hope is to find some approximation algorithms. We first give a very simple approximation algorithm. Then we give a slightly more involved algorithm with a better approximation performance.

Theorem 16 *There is a ρ-approximation algorithm for the Integer Max-Flow WSNC problem, where $\rho = \max_{i \in N} \frac{\max_{ij \in A} e_{ij}}{\min_{ij \in A} e_{ij}}$.*

PROOF. Convert the sensor network into a flow network as follows. We give the edges unbounded capacity and we give each node i the capacity $c_i = \lfloor \frac{E_i}{\max e_{ij}} \rfloor$. Then a standard Max-Flow Min-Cut algorithm with node capacities will yield a polynomial time algorithm that is at worst a ρ-factor from the optimum. □

Unfortunately the above approximation is not of constant ratio. Note that for the fixed range model where each sensor has a constant number of fixed power

settings, the above algorithm does give a constant ratio approximation.

A better approximation algorithm is the following. The key idea is to first solve the fractional LP-formulation in polynomial time, and then try to find a large integer flow that is close to the optimum value.

Theorem 17 *There is a polynomial time approximation algorithm for the Integer Max-Flow WSNC problem, which computes an integer maximum flow with value $F_{approx} \geq F_{optimum} - m$, where m is the number of arcs of the graph.*

PROOF. We give the proof for the case when there is only one source. It is a simple exercise to generalize the proof to the case with multiple sources.

First, we solve the relaxation of the problem optimally, i.e., we allow flows to be of real value. As this is an LP, the ellipsoid method gives us in polynomial time an optimal solution F^*, that can be realized within the energy constraints.

We now find a large integer flow inside F^* in the following way. We use an integer flow function F, which invariantly will map each arc ij to a non-negative integer f_{ij} with $f_{ij} \leq f^*_{ij}$, and which has conservation of flows; i.e., F will invariantly be a flow that fulfills the energy constraints. Initially, set F to be 0 on all arcs.

Now, repeat the following step while possible. Find a path P from s to t, such that for each arc ij on the path, $f^*_{ij} - f_{ij} \geq 1$. Let $f_P = min_{ij \in P} \lfloor f^*_{ij} - f_{ij} \rfloor$ be the minimum over all arcs ij on the path P. Note that $f_P \geq 1$. Now add f_P to each f_{ij} for all arcs ij on the path P. Observe that the updated function F fulfills the energy constraint conditions.

The process ends when each path from s to t contains an arc with $f^*_{ij} - f_{ij} < 1$. We note that $F^* - F$ is a flow, and standard flow techniques show that its value is at most m, the number of arcs. (For let S be the set of nodes, reachable from s by a path with all arcs fulfilling $f^*_{ij} - f_{ij} \geq 1$. Since $t \notin S$, $(S, V - S)$ is a cut and its capacity with respect to the flow $F^* - F$ is at most the number of arcs across the cut.) Since the value of F^* is at least $F_{optimum}$, and hence the value of F is at least $F_{optimum} - m$.

In each step of the procedure given above, we obtain at least one new arc ij with $f^*_{ij} - f_{ij} < 1$; this arc will no longer be chosen in a path in a later step. Thus, we perform at most m steps. Each step can be done easily in linear time. Hence, the algorithm is polynomial, using the time of solving one linear program plus $O(m(n + m))$ time for computing the approximate flow. □

This algorithm is still not of constant ratio but we conjecture that it is a step towards a 2-approximation algorithm.

4 Graphs with Bounded Treewidth

Many hard graph problems have polynomial (sometimes even linear) time algorithms when restricted to graphs of bounded treewidth. This however is not the case for the Integer Max-Flow WSNC problem: the problem remains hard. The treewidth parameter nicely delineates the classes of graphs for which the Integer Max-Flow WSNC is of apparently increasing complexity problem in the following manner.

- *Graphs of treewidth one:* These are just *forests* and have a very simple linear time algorithm: remove all nodes that do not have a path to the sink t; compute in the resulting tree in post-order for each node the number of packets it receives from its children and then the number of packets it can send to its parent.
- *Graphs of treewidth two:* We show below that if there is a single source, and the graph with an edge added between this single source and the sink node has treewidth two, then then there is a polynomial time algorithm for the Integer Max-Flow WSNC problem. The general case remains open.
- *Graphs of bounded treewidth greater than two:* We show that in this case, the problem is *weakly* NP-complete. We give a pseudo-polynomial time algorithm for this class.
- *Graphs of unbounded treewidth:* In this case, the problem is *strongly* NP-complete and even APX-hard, as shown in the previous section.

4.1 Graphs of Treewidth Two

In this section, we will show that the Integer Max-flow WSNC problem (with one source and one sink) is polynomial-time solvable if the treewidth of the graph obtained by adding an edge from the source to the sink is bounded by two. While this is a somewhat specific case, we think this case is interesting because it partially bridges the gap between the trivial case of trees and the NP-hard case of treewidth three, and because the algorithmic technique is of some interest: unlike most algorithms that exploit treewidth, it does not use dynamic programming but a reduction strategy.

For a simple description of the algorithm, we generalize our problem in two ways. First, we allow *parallel* arcs. As a consequence, we need to change our notation somewhat, and denote an arc with its name, instead of by the pair of endpoints. Parallel arcs may require a different energy per packet that is transmitted across them. Secondly, we assume that each arc p has a capacity c_p. The capacity of an arc is a positive integer and denotes the maximum number of packets that can be transmitted across the arc.

If we have an instance without capacity, we can set for each arc $p = ij$, $c_p = \lfloor E_i/e_{ij} \rfloor$. Arcs with zero capacity can be removed.

Given a directed graph $G = (N, A)$, with a source s and a sink t, we build the undirected graph $G^u = (N, A^u)$ as follows:

- There is an edge in A^u between i and j, if and only if there is at least one arc from i to j and/or at least one arc from j to i in A.
- We add an edge from s to t.

G_u will be used as an auxiliary graph in subsequent constructions.

Theorem 18 *The Integer Max-flow WSNC problem, with parallel arcs and arc capacities and with a single source s and sink t, can be solved in $O(m \log \Delta)$ time on directed graphs $G = (N, A)$, such that the graph $(N, A \cup \{st\})$ has treewidth at most two, where Δ is the maximum outdegree of a node in G, and $m = |A|$.*

Our algorithm is based upon the principle of *reduction*. While most algorithms that solve problems on graphs of small treewidth are based upon dynamic programming, some algorithms are also based upon reduction. (See e.g. [2,7].)

We first need a result on graphs with small treewidth.

Lemma 19 (Ramachandramurthi [19]) *If G^u is a simple, undirected graph of treewidth at most k that is not a clique, then G^u has two non-adjacent vertices of degree at most k.*

Corollary 20 *If G^u is a simple, undirected graph of treewidth at most two, with at least three vertices, then there is a vertex of degree at most two that is neither the source s nor the sink t.*

PROOF. If G^u is a clique, then it has exactly three vertices, and the result trivially holds. Otherwise, let i and j be the two non-adjacent vertices of degree at most two, as indicated by Lemma 19. At least one of these two vertices is unequal to s and to t. □

Now, we can repeat the following step. We first build G^u. Then, we find a vertex i in G^u that is neither the source s nor the sink t, and that has degree at most two in G^u (cf. Corollary 20). We now apply a *reduction* step, that transforms G to an equivalent network without i, i.e., the number of vertices is decreased by one. We first describe the reduction step, and then will discuss a more efficient implementation.

Let i be a vertex in G^u that is unequal to s and t, and that has degree at most two.

If i has degree 0 then clearly we can remove i from G, and obtain an equivalent network.

Suppose now that i has degree one in G^u. This means that there is a vertex j, such that all arcs in G with i as tail have j as head, and all arcs with i as head have j as tail. Note that there always is an optimal flow that does not transmit any packet from and to i. Thus, we can remove i and all arcs that have i as one of its endpoints.

We now look at the most interesting case, namely that i has degree exactly two in G^u. Suppose the neighbors of i in G^u are j and k. The arcs with i as one of its endpoints are of the form:

- from j to i,
- from i to k,
- from k to i,
- from i to j.

Call the arcs of the form ji and ik *forward arcs*, and arcs of the form ki and ij *backward arcs*. Consider a flow with maximum value that has as additional condition that the total energy used by all nodes is minimal. In such a flow, either no packets will be transmitted over the forward arcs, or no packets will be transmitted over the backward arcs. (If both packets are transmitted over forward and backward arcs, then we can cancel some and obtain a feasible flow with the same value but smaller total energy.) For this reason, we can handle forward and backward arcs independently.

First, consider all the forward arcs. We first compute a bound on the number of packets that i can transmit to k, as follows. Suppose there are b number of arcs ik. Sort these in order of non-decreasing energy per packet. Let the arcs have energy per packet $e_1 \leq e_2 \leq \cdots \leq e_b$, and the corresponding capacities of the arcs $c_1, c_2, \ldots, c_b$. For all q, $1 \leq q \leq b$, there is always an optimal flow that only transmits a packet on the q^{th} arc, when all p^{th} arcs, with $p < q$ have totally used up their capacity. Thus, we can compute a bound C_{ik} on the number of packets that i can transmit to k as follows.

If $\sum_{p=1}^{b} e_p \cdot c_p \leq E_i$, then take $C_{ik} = \sum_{p=1}^{b} c_p$. Otherwise, suppose that

$$\sum_{p=1}^{q} e_p \cdot c_p \leq E_i < \sum_{p=1}^{q+1} e_p \cdot c_p$$

i.e., we have sufficient energy to use all the capacity of the first q arcs ik, but not for the first $q+1$ arcs. We thus use all capacity of these first q arcs, and possibly some part of the capacity of the $(q+1)^{th}$ arc. Note that, after we used up all the capacity of the first q arcs, we only have energy left to transmit at

most

$$\left\lfloor \left(E_i - \sum_{p=1}^{q} e_p \cdot c_p \right) / e_{q+1} \right\rfloor$$

packets. Thus, we set

$$C_{ik} = \sum_{p=1}^{q} e_p \cdot c_p + \left\lfloor \frac{E_i - \sum_{p=1}^{q} e_p \cdot c_p}{e_{q+1}} \right\rfloor .$$

Observe that when C_{ik} packets arrive at i, they all can be forwarded to k, but we can never transmit more than C_{ik} packets from i to k.

In the reduction, we build a new graph G', where we have removed i and its incident arcs, and added possibly a number of arcs between j and k (possibly in both directions). G' has vertex set $N - \{i\}$, and each arc in G that does not involve i is also an arc in G'. The arcs of the form jk in G' are obtained as follows.

Suppose there are d arcs of the form ji in G. Sort these in order of non-decreasing energy, and suppose these arcs have energies $e_1 \leq e_2 \leq \cdots \leq e_d$, with corresponding capacities $c_1, c_2, \cdots, c_d$. Again, we may assume that we transmit only packets over the arc with energy e_q if we have used up all capacities over all arcs with energy e_p, $p < q$.

If $\sum_{p=1}^{d} c_p \leq C_{ik}$, then we replace each arc of the form ji in G by an arc of the form jk with the same energy cost and capacity: all packets transmitted through these arcs can be forwarded to k, so we have the same number of packets that can go from j to k, using the same amount of energy at j. Otherwise, suppose that for some node i with $0 \leq i < s$,

$$\sum_{p=1}^{q} c_p \leq C_{ik} < \sum_{p=1}^{q+1} c_p \, .$$

By the assumption made above, it follows that we will not be transmitting packets over the arcs with energy larger than e_{q+1}, and do not use the full capacity of the $(q+1)^{th}$ arc. Thus, in G' we take an arc jk with capacity c_p and energy cost e_p for each $p \leq q$, and an arc with capacity

$$c'_{q+1} = C_{ik} - \sum_{p=1}^{q} c_p$$

and energy cost e_{q+1}. (If $c'_{q+1} = 0$, we do not take the arc.) Note that the total capacity of the arcs jk is now C_{ik}. It is not hard to see that we can transmit the same number of packets in G from j to k via i as over the new arcs jk in G'.

For the backward arcs, we perform exactly the same step, except that the roles

of j and k are switched. In this fashion, we obtain an equivalent network, but now with one less vertex.

This finishes the description of the reduction step. It is straightforward to see that this step can be done in polynomial time. Different data structures may help to speed up the performance of the reduction step.

Here, we use the *mergeable heap* data structure. This data structure performs the following operations on a collection of elements. Each element has a *key* and possibly more information stored. Our collection is partitioned into disjoint sets. In our data structure, we can create a new set with one new element, with a key value and possibly other information; for a given set, obtain a pointer to the object that represents an element with *maximum* key value (and thus obtain this maximum, and possibly update the other information); for a given set, we can delete this element with maximum key value; and for two given sets, we can take the *disjoint union* of the two sets, i.e., replace these sets by their disjoint union. Stylized implementations of mergeable heaps include the so called binomial heaps and the Fibonacci heaps, see e.g., [11, Chapters 19 and 20].

Lemma 21 *There is a data structure that allows unions of sets, obtaining the element with maximum key, deleting the element with maximum key, and creating new one element sets, such that each operation takes at most $O(\log d)$ time, where d is the maximum size of any set that is built.*

Note that we cannot expect a much faster data structure, otherwise we can sort d elements using $O(d)$ operations on the data structure; create a set for each element, then take the union of all sets and then iteratively obtain and delete the element with maximum key until no elements are left. The order in which the elements are deleted is sorted, from largest till smallest. Thus, $O(d)$ operations on the data structure have to cost $\Omega(d \log d)$ time, or $\Omega(\log d)$ in the worst case.

We now describe how the mergeable-heap data structure can be used to obtain an overall time of $O(m \log \Delta)$, with Δ the maximum outdegree of a node in G.

Note that the outdegree will never increase during a reduction step, and hence at each point in the algorithm, each node has an outdegree that is at most Δ.

We use a mergeable-heap data structure, with an element for each arc. For each node i and j, we have a set with all arcs from i to j, if there is at least one such arc. We also maintain the graph G^u. As keys we use the cost of arcs; each arc has also its capacity stored. We further store the total capacity of all arcs from i to j; we denote this value by C'_{ij}.

Now, a reduction step can be carried out as follows. We only consider the forward arcs as the backward arcs are similar. We use i, j, k, b, and d as in the description of the reduction step for nodes of degree two in G^u, i.e., we look at the arcs ji and ik, and obtain arcs jk.

The procedure has three main steps.

First, we consider the arcs from i to k, and compute the value C_{ik}, as described above. It is not hard to see that we can do this in $O(d \log d)$ time, if there are d such arcs. As each of the arcs from i to k can be deleted after this step, and i has outdegree at most Δ, the total time of this step over all reductions is bounded by $O(m \log \Delta)$.

The second step is only carried out if $C'_{ij} > C_{ik}$. For a faster implementation, we do not sort the arcs from j to i, but instead we delete and/or update the capacities of the arcs from j to i in order of non-increasing cost.

We repeat the following step, till $C'_{ji} \leq C_{ik}$. Obtain from the mergeable-heap data structure an arc from j to i with maximum cost e_{ji}. Using the same notation as above, suppose this is the p^{th} arc, with cost e_p and capacity c_p. If $C'_{ij} - c_p \geq C_{ik}$, then we delete this arc with maximum cost, and decrease C'_{ij} by c_p. This operation can be done in $O(\log \Delta)$ time on the mergeable-heap data structure. If $C'_{ij} - c_p < C_{ik}$, then we must update the capacity of the arc: its new capacity must be $C_{ik} - C_{ji} + c_p$, as $C_{ji} - c_p$ is the sum of the capacities of the first $p-1$ arcs from j to i (in order of non-decreasing cost). We also set C_{ji} to C_{ik}.

One can verify that this step indeed gives the same collection of arcs and capacities as the procedure described above where we first sorted the arcs in order of non-decreasing cost. Note that all but possibly one of the forward arcs that we considered are permanently deleted. Thus, if we delete d arcs in a reduction step, the step can be carried out in $O((d+1)\Delta)$ time, which amounts to a total of $O(m \log \Delta)$.

In the third step, all arcs from j to i now become arcs from j to k. We do not need to update this information for each arc, as it is sufficient if each set *knows* the endpoints of the arcs it represents. The third step has two cases. If before the reduction, there was a data structure with arcs from j to k, then we take the union of the data structure for arcs ji and arcs jk: these arcs now all are arcs from j to k. If there were no arcs from j to k before the reduction, then the data structure for the arcs ji now acts as data structure for arcs jk.

The time for this third step thus is dominated by the time for one union in the mergeable-heap data structure, i.e., it costs $O(\log \Delta)$ per reduction. As we perform $O(n)$ reduction, the total time over all third steps is $O(n \log \Delta)$.

Thus, the total time is bounded by $O(m \log \Delta)$. This completes the proof of Theorem 18.

The problem for treewidth two graphs with multiple sources is still open. If we apply the construction from Section 4.2 that transforms a graph with treewidth three with multiple sources to a graph with one source, then in many cases, the treewidth grows to three. So the Integer Max-Flow WSNC problem for general graphs of treewidth two remains open.

4.2 Graphs of Bounded Treewidth Greater than Two

Our second result on treewidth is that the Integer Max-Flow WSNC problem is NP-hard for graphs of treewidth three. This shows that a result like the previous one for treewidth two graphs cannot be found when the treewidth is three or more.

Theorem 22 *The Integer Max-Flow WSNC problem is NP-hard for graphs of treewidth three.*

The proof resembles the proof of strong NP-hardness for general graphs, but we use here the (weak) NP-complete problem 2-PARTITION instead of the strong NP-complete problem 3-PARTITION.

2-PARTITION
INSTANCE: Given a multiset $\mathcal{S}$ of n positive integers $a_1, \ldots, a_n$ and a positive integer $B = \sum_{i=1}^{n} a_i/2$.
QUESTION: Can $\mathcal{S}$ be partitioned into two subsets each of equal sum B.

PROOF. Given an instance I of 2-PARTITION, we create an instance I' of the Integer Max-Flow WSNC problem as follows.

Each number a_i represents a sensor node w_i each with energy $E_i = B$. Additionally, we have a source node s and sink node t along with two special nodes v_1 and v_2, each of them with energy B. Now connect each v_j $(j = 1, 2)$ to each w_i with arc cost $e_{ji} = a_i$. The source node s is connected to v_1 and v_2 with arc cost 1 each, and each node w_i is connected to the sink node t with cost B.

Now, there is an energy constrained flow with n packets from s to t, if and only if $a_1, \ldots, a_n$ can be partitioned into two subsets, each of sum B.

If $a_1, \ldots, a_n$ can be partitioned into two subsets $\mathcal{S}_1$, $\mathcal{S}_2$, each of sum B, then we can build a flow as follows: s transmits $|\mathcal{S}_1|$ packets to v_1 and $|\mathcal{S}_2|$ packets to v_2. For each $a_i \in \mathcal{S}_1$, v_1 transmits a packet to w_i, and similarly for each $a_i \in \mathcal{S}_2$, v_2 transmits a packet to w_i, each packet consuming a_i energy from

each node. Finally, each w_i now transmits one packet to t. This fulfills the requirements of transmitting n packets from s to t.

Suppose now there is an energy constrained maximum flow of n packets from s to t. Note that each w_i can transmit at most one packet to t, and as there are n packets to t, each w_i must transmit exactly one packet to t. So, for each i, either v_1 or v_2 transmits a packet to w_i. If v_1 transmits a packet to w_i, then put a_i in $\mathcal{S}_1$, otherwise put a_i in $\mathcal{S}_2$. It is clear that $\mathcal{S}_1$ and $\mathcal{S}_2$ partition $\mathcal{S}$. For each element $a_i \in \mathcal{S}_1$, v_1 must transmit a packet with cost a_i, hence the sum of the elements in $\mathcal{S}_1$ is at most B. Similarly, the sum of the elements in $\mathcal{S}_2$ is at most B, and as the sum of all a_i's equals $2B$, the sum of all elements in $\mathcal{S}_1$ equals B, and likewise for $\mathcal{S}_2$. □

The constructed graph has a feedback vertex set of size two: if we remove v_1 and v_2 of the graph, we obtain a forest. Thus, it has treewidth at most three (see e.g. [5]). Its treewidth is also at least three, as it contains the complete graph K_4 as a minor (remove $w_3, \ldots, w_n$, contract s to v_1, and w_2 to t). Hence it has treewidth exactly three. The result also rules out a polynomial time algorithm for the Integer Max-Flow WSNC problem on graphs of bounded treewidth, unless P=NP.

However, we show now that there is a pseudo-polynomial time algorithm whenever the treewidth is bounded.

Theorem 23 *The Integer Max-Flow WSNC problem can be solved in pseudo-polynomial time on graphs of bounded treewidth.*

PROOF. We begin with a number of auxiliary definitions. A *boundary directed graph* is a triple (N, A, X), with (N, A) a directed graph, and $X \subseteq N$ a set of distinguished vertices, called the *boundary*.

Suppose (N', A', X) is a boundary directed graph with (N', A') a subgraph of the given graph $G = (N, A)$, with given costs and energies, and source s and sink t.

A *partial energy constrained flow*, or in short *pecf*, in this boundary directed graph is a function that assigns to each arc ij in A' a flow-value, such that

- for all $i \in N' - \{s, t\}$, the inflow of i equals the outflow of i, and
- for all $i \in N'$, the cost of the outflow of i is at most E_i.

The *signature* of a *pecf* is a pair (q, ϵ), with q a function that maps each $i \in X$ to the outflow of i minus the inflow of i:

$$q(i) = \sum_{ij \in A} f_{ij} - \sum_{ji \in A} f_{ji}$$

and ϵ a function that maps each $i \in X$ to the used energy at sensor i:

$$\epsilon(i) = \sum_{ij \in A} f_{ij} \cdot e_{ij} \,.$$

For ease of notation, we also use f_{ij} when there is no arc $ij \in A$; in such a case, $f_{ij} = 0$. The algorithm we will design uses a special type of tree decomposition: a *nice tree decomposition* where, in addition to the usual requirements, we assume that the root of the tree decomposition r has a bag that contains only s: $X_r = \{s\}$. A nice tree decomposition has four types of nodes:

- **Leaf** nodes: node α which is a leaf of the tree with $|X_\alpha| = 1$.
- **Introduce** nodes: node α with one child β with $X_\alpha = X_\beta \cup \{i\}$ for some node i.
- **Forget** nodes: node α with one child β, with $X_\alpha = X_\beta - \{i\}$ for some node i.
- **Join** nodes: node α with two children β, γ with $X_\alpha = X_\beta = X_\gamma$.

For each node α in the tree decomposition, we define a boundary directed graph $G_\alpha = (N_\alpha, A_\alpha, X_\alpha)$, with N_α the set of all vertices in X_α or X_β with β a descendant of α, and A_α the set of all arcs in A between vertices in N_α.

A small modification of the construction in [6] shows that we can obtain in linear time, given a tree decomposition of width at most k of a graph G, a nice tree decomposition of G of width at most k, such that the root node r has $X_r = \{s\}$.

Our algorithm mainly consists of iteratively computing for each node α in the tree decomposition a table consisting of all signatures of all pecf's in G_α. The tables are computed in post-order, i.e., we compute a table for a node when the tables of its children are known.

Note that the size of each table is pseudo-polynomial, as there are pseudo-polynomially many signatures of pecf's in G_α: for each $i \in X_\alpha$, $\epsilon(i)$ is an integer between 0 and E_i, the outflow of i is an integer between 0 and E_i, the inflow of i is an integer between 0 and $\sum_{ji \in A} E_j$, and hence $q(i)$ is an integer between $-\sum_{ji \in A} E_j$ and E_i.

We now give for each of the four types of nodes a description of a procedure that computes for a node α the table of all signatures of all pecf's in G_α, given such tables for the children of α.

Leaf nodes: It is trivial to see that the table can be computed directly for a **Leaf** node.

Introduce nodes: Suppose α is an **Introduce** node with β the unique child of α and $X_\alpha = X_\beta \cup \{i\}$. See Algorithm 1. For each element $(q^\sigma, \epsilon^\sigma)$ from the table of signatures of pecf's in G_β, the algorithm does the following. It enumerates all assignments of flow values to all arcs $ij \in A$ and $ji \in A$, where f_{ij} is an integer between 0 and E_i, and f_{ji} is an integer between 0 and E_j. For each of these assignments of flow values, it computes a corresponding signature for a pecf in G_α, checks if no vertex in X_α uses too much energy, and if so, stores it in the table for α.

Algorithm 1 Computation for **Introduce** nodes

for each signature $(q^\sigma, \epsilon^\sigma)$ for β **do**
  for each assignment $f_{ij} \in \{0, \ldots, E_i\}$ for all $j \in X_\beta$ with $ij \in A$ **do**
    for each assignment $f_{ji} \in \{0, \ldots, E_j\}$ for all $j \in X_\beta$ with $ji \in A$ **do**
      {determine a signature $\sigma' = (q', \epsilon')$}
      $\epsilon^{\sigma'}(i) = \sum_{j \in X_\beta} f_{ij} \cdot e_{ij}$
      for all $j \in X_\beta$ **do**
        $\epsilon^{\sigma'}(j) = \epsilon^\sigma(j) + f_{ji} \cdot e_{ji}$
      end for
      $q^{\sigma'}(i) = \sum_{j \in X_\beta} f_{ij} - \sum_{j \in X_\beta} f_{ji}$
      for all $j \in X_\beta$ **do**
        $q^{\sigma'}(j) = q^\sigma(j) + f_{ji} - f_{ij}$
      end for
      if for all $j \in X_\alpha$: $\epsilon^{\sigma'}(j) \leq E_j$ **then**
        store σ' in the table of signatures for α
      end if
    end for
  end for
end for

Lemma 24 *Algorithm* 1 *correctly computes the table of signatures of all pecf's in* G_α *for an* **Introduce** *node* α.

PROOF. Suppose g is a pecf in G_α. Let g' be the pecf in G_β, obtained by restricting g to the arcs in G_β. Consider the iteration in Algorithm 1, where $f_{ij} = g_{ij}$ and $f_{ji} = g_{ji}$ for all $j \in X_\beta$, and where $(q^\sigma, \epsilon^\sigma)$ is the signature of g'. It is not hard to verify that we need to store the signature of g in this iteration, if the energy constraint is satisfied. So, we store the signature of each pecf in G_α.

Suppose we store a signature σ' in the table, and suppose this is in the iteration where we use signature $(q^\sigma, \epsilon^\sigma)$ for β, and values f_{ij}, f_{ji} for all $j \in X_\beta$. Let g' be the pecf in G_β with signature σ. Let g be the function, obtained by taking $g_{ij} = f_{ij}$, $g_{ji} = f_{ji}$ for all $j \in X_\beta$, and $g_{j\ell} = g'_{j\ell}$ for all arcs $j\ell$ in G_β. One can verify that σ' is the signature of g. Thus, the element in the table is the signature of a pecf in G_α. $\square$

Forget nodes: Suppose α has one child β, with $X_\beta = X_\alpha - \{i\}$. Note that $i \neq s$, as the root of the tree decomposition contains s. See Algorithm 2.

Algorithm 2 Computation for **Forget** nodes

```
for each signature (q^σ, ε^σ) for β do
  if q^σ(i) = 0 or i = t then
    {determine a signature σ' = (q', ε')}
    for all j ∈ X_α do
      q^σ'(j) = q^σ(j)
      ε^σ'(j) = ε^σ(j)
    end for
    store σ' in the table of signatures for α
  end if
end for
```

Note that G_α and G_β have the same vertex and arc sets. As the boundary of G_α is a subset of the boundary of G_β, each pecf in G_α is a pecf in G_β. A pecf in G_β is also a pecf in G_α, if and only if the inflow for i equals the outflow or $i \in \{s, t\}$. This is because i is a new internal vertex. Because the root of the tree decomposition contains s, we have $i \neq s$, and the condition can be stated as $q^\sigma(i) = 0$ or $i = t$. If this condition holds, Algorithm 2 stores the restriction of the signature to the nodes in X_α in the table of signatures for α. Thus:

Lemma 25 *Algorithm 2 correctly computes the table of signatures of all pecf's in G_α for a* **Forget** *node α.*

Join nodes: Suppose α has two children β and γ, with $X_\alpha = X_\beta = X_\gamma$.

Below, we assume a function to be 0 for all elements outside its domain.

Lemma 26 *Let f be a function, mapping each arc in G_α to a non-negative integer. Then f is a pecf in G_α, if and only if there is a pecf f' in G_β and a pecf f'' in G_γ, with*

(1) $f = f' + f''$, and
(2) for all $i \in X_\alpha$, $\sum_{ij \in A,\ j \in N_i} f_{ij} \leq E_i$.

PROOF. Suppose f is a pecf in G_α. Let f' be the restriction of f to the arcs in G_β. Let f'' be obtained by taking $f''_{ij} = 0$ if $i \in X_\alpha$ and $j \in X_\alpha$, and $f''_{ij} = f_{ij}$ for all arcs in G_β with at least one endpoint not in X_α. One can verify that f' is a pecf in G_β, and f'' is a pecf in G_γ. Clearly $f = f' + f''$. The last stated condition follows directly, as f is a pecf in G_α.

Now, suppose we have a pecf f' in G_β and a pecf f'' in G_γ that fulfill the two conditions. Note that when the inflow equals the outflow for a node i in f' and in f'', it also does so in $f = f' + f''$. So, the first condition of a pecf holds, and the second condition holds by assumption. □

We now compute the table of the signatures of the pecf's in G_α. See Algorithm 3.

Algorithm 3 Computation for **Join** nodes

```
for each signature (q^σ, ε^σ) for β do
  for each signature (q^σ', ε^σ') for γ do
    {determine a signature σ' = (q', ε')}
    for all i ∈ X_α do
      ε^σ''(i) = ε^σ(i) + ε^σ'(i)
      q^σ''(i) = q^σ(i) + q^σ'(i)
    end for
    if for all i ∈ X_α: ε^σ''(i) ≤ E_i then
      store σ'' in the table of signatures for α
    end if
  end for
end for
```

We now establish its correctness.

Lemma 27 *Algorithm* 3 *correctly computes the table of the signatures of pecf's in* G_α *for a* **Join** *node* α.

PROOF. One can verify that if σ is the signature of a pecf f' in G_β, and σ' is the signature of a pecf f'' in G_γ, then the signature σ'' as computed by Algorithm 3 is the signature of $f' + f''$. Correctness follows now directly from Lemma 26. □

We now complete the proof of Theorem 23. Using the procedures for the different types of nodes, we can compute all tables, in post-order. As each table has pseudo-polynomial size, and the time per table is polynomial in the number of signatures of the tables of the children for **Join**, **Introduce** and **Forget** nodes, and $O(1)$ for **Leaf** nodes, this costs pseudo-polynomial time.

Lemma 28 *F packets can be transmitted from s to t in the network, if and only if the table of the root r of the tree decomposition contains a signature (q, ϵ) for some ϵ with $q(s) = F$.*

PROOF. Note that G_r has the same vertices and arcs as G. Thus, a pecf in G is a flow in G that fulfills the energy constraint, and vice versa. The value of a flow with signature (q, ϵ) equals $q(s)$. Now the lemma follows. □

After all tables have been computed, we can find the value of the optimal flow by inspecting the table of the root node: by Lemma 28, this value equals the maximum F, such that a signature (q, ϵ) for some ϵ with $q(s) = F$ belongs to the table of the root of the tree decomposition. With additional bookkeeping, one can also find the corresponding flow. □

5 Conclusion

The Maximum Flow WSNC problem is an interesting and relevant problem, with practical implications in the context of wireless ad hoc networks. In this paper, we studied the integer variant of the problem. We obtained a good polynomial time approximation algorithm for the problem, which is however not of constant performance ratio. We also studied how the complexity of the problem depends on the treewidth of the network. We found that except for the case where each sensor has one fixed power setting or when the underlying graph is of treewidth two with an edge joining the source and sink nodes, the problem is weakly NP-complete for bounded treewidth greater than two. It is strongly NP-complete for networks of unbounded treewidth and in fact even APX-hard. It is also strongly NP-complete on geometric configurations on a line.

Acknowledgements

We thank Alex Grigoriev, Han Hoogeveen, Arie Koster, Erik van Ommeren and Gerhard Woeginger for fruitful discussions and helpful comments.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 1993.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. ACM*, 40:1134–1164, 1993.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Berlin, 1999.

[4] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[5] H. L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comp. Sc.*, 209:1–45, 1998.

[6] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

[7] H. L. Bodlaender and B. van Antwerpen-de Fluiter. Reduction algorithms for graphs of small treewidth. *Information and Computation*, 167:86–119, 2001.

[8] J. Chang and L. Tassiulas. Fast approximate algorithms for maximum lifetime routing in wireless ad-hoc networks. In *Proceedings Networking 2000*, pages 702–713, 2000.

[9] J. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transaction on Networking*, 12(4):609–619, 2004.

[10] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.*, 35(3):713–728, 2005.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, Cambridge, Mass., USA, 2001.

[12] P. Floréen, P. Kaski, J. Kohonen, and P. Orponen. Exact and approximate balanced data gathering in energy-constrained sensor networks. *Theor. Comp. Sc.*, 344(1):30–46, 2005.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[14] N. Garg and J. Könemann. Faster and simpler algorithms for multi-commodity flow and other fractional packing problems. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[15] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings 33rd Hawaii International Conference on System Sciences HICSS-33*, 2000.

[16] B. Hong and V. K. Prasanna. Maximum lifetime data sensing and extraction in energy constrained networked sensor systems. *J. Parallel and Distributed Computing*, 66(4):566–577, 2006.

[17] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks*, 42(6):697–716, 2003.

[18] F. Ordonez and B. Krishnamachari. Optimal information extraction in energy-limited wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 22(6):1121–1129, 2004.

[19] S. Ramachandramurthi. The structure and number of obstructions to treewidth. *SIAM J. Disc. Math.*, 10:146–157, 1997.

[20] Y. Xue, Y. Cui, and K. Nahrstedt. Maximizing lifetime for data aggregation in wireless sensor networks. *Mobile Networks and Applications*, 10:853–864, 2005.

[21] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufman, 2004.