Constrained Flow Optimization with Applications to Data Gathering in Sensor Networks*

Bo Hong and Viktor K. Prasanna
University of Southern California, Los Angeles CA 90089-2562
{bohong, prasanna}@usc.edu

Abstract

We focus on data gathering problems in energy-constrained wireless sensor networks. We study store-and-gather problems where data are locally stored on the sensors before the data gathering starts, and continuous sensing and gathering problems that model time critical applications. We show that these problems reduce to maximization of network flow under vertex capacity constraint, which reduces to a standard network flow problem. We develop a distributed and adaptive algorithm to optimize data gathering. This algorithm leads to a simple protocol that coordinates the sensor nodes in the system. Our approach provides a unified framework to study a variety of data gathering problems in sensor networks. The efficiency of the proposed method is illustrated through simulations.

1 Introduction

A major application of the sensor networks is to periodically monitor the environment, such as vehicle tracking and classification in the battle field, patient health monitoring, pollution detection, etc. The basic operation in such applications is to sense the environment and eventually transmit the sensed data to the base station for further processing. The data gathering processes are usually implemented using multi-hop packet transmissions, since short range communications can save energy and also reduce communication interferences in high density WSNs [1].

Although the applications may be designed for different functionalities, the underlying data gathering processes share a common characteristic: compared with sensing and computing, transferring data between the sensors is the most expensive (in terms of energy consumption) operation [1]. In this paper, we study the energy efficiency of data gathering in the WSNs from an algorithmic perspective. By modeling the energy consumption associated with each send and receive operation, we formulate the data gathering problem as a constrained network flow optimization problem where each each vertex u is associated with a capacity constraint w_u, so that the total amount of flow going through u (incoming plus outgoing flow) does not exceed w_u. We show that such a formulation models a variety of data gathering problems (with energy constraint on the sensor nodes).

*Supported by the National Science Foundation under award No. IIS-0330445 and in part by DARPA under contract F33615-02-2-4005.

CENG 2004-07

Some variations of the data gathering problem have been studied recently. In [6], the data gathering is assumed to be performed in *rounds* and each sensor can communicate (in a single hop) with the base station and all other sensors. The total number of rounds is then maximized under a given energy constraint on the sensors. In [9], a non-linear programing formulation is proposed to explore the trade-off between energy consumed and the transmission bandwidth, which models the radio transmission energy according to Shannon's theorem. In [10], the data gathering problem is formulated as a linear programing problem and a $1 + \omega$ approximation algorithm is proposed. This algorithm further leads to a distributed heuristic.

Our study departs from the above with respect to the problem definition as well as the solution technique. For short-range communications, the difference in the energy consumption between sending and receiving data packets is almost negligible. We adopt the reasonable approximation that sending a data packet consumes the same amount energy as receiving a data packet [1]. The study in [9] and [10] differentiate the energy dissipated for sending and receiving data. Although the resulting problem formulations are indeed more accurate than ours, the improvement in accuracy is marginal for short-range communications.

In [6], each sensor generates exactly one data packet per round (a round corresponds to the occurrence of an event in the environment) to be transmitted to the base station. The system is assumed to be fully connected. The study in [6] also considers a very simple model of data aggregation where any sensor can aggregate all the received data packets into a single output data packet. In our sensor network model, each sensor communicates with a limited number of neighbors due to the short range of the communications, resulting in a general graph topology for the system. We study store-and-gather problems where data are locally stored on the sensors before the data gathering starts, and continuous sensing and gathering problems that models time critical applications. A unified flow optimization formulation is developed for the two classes of problems.

The constrained flow problem reduces to the standard network flow problem, which is a classical flow optimization problem. Many efficient algorithms have been developed ([2]) for the standard network flow problem. However, in terms of decentralization and adaptation (two properties desirable for WSNs), these well known algorithms are not suitable for data gathering problems in WSNs. In this paper, we develop a decentralized and adaptive algorithm for the maximum network flow problem. This algorithm is a modified version of the Push-Relabel algorithm [3]. In contrast to the Push-Relabel algorithm, it is adaptive to changes in the system. It finds the maximum flow in $O(n^2 \cdot |V|^2 \cdot |E|)$ time, where n is the number of adaptation operations, $|V|$ is the number of nodes, and $|E|$ is the number of links. With the algorithm thus established, we further develop a simple distributed protocol for data gathering. The performance of this protocol is studied through simulations.

The rest of the paper is organized as follows. The data gathering problems are discussed in Section 2. We show that these problems reduce to network flow problems with constraint on the vertices. In Section 3, we develop a mathematical formulation of the constrained network flow problem and show that it reduces to a standard network flow problem. In Section 4, we derive a relaxed form for the network flow problem. A distributed and adaptive algorithm is then developed for this relaxed problem. A simple protocol based on this algorithm is presented in Section 4.3. Experimental results are presented in Section 5. Section 6 concludes this paper.

2 Data Gathering Problems in Sensor Networks with Energy Constraint

2.1 Sensor Network Model

Suppose a network of sensors is deployed over a region. The location of the sensors are fixed and known *a priori*. The sensor network is represented by a graph $G(V, E)$, where V is the set of sensor nodes. $(u, v) \in E$ if $u \in V$, $v \in V$ and u is within the communication range of v. σ_u, the set of successors of u, is defined as $\sigma_u = \{v \in V | (u, v) \in E\}$. Similarly, ψ_u, the set of predecessors of u is defined as $\psi_u = \{v \in V | (v, u) \in E\}$. The event is sensed by a subset of sensors $V_c \subset V$. r is the base station to which the sensed data are transmitted. Sensors $V - V_c - \{r\}$ in the network does not sense the event but can relay the data sensed by V_c.

Data transfers are assumed to be performed via multi-hop communications where each hop is a short-range communication. This is due to the well known fact that long-distance wireless communication is expensive in terms of both implementation complexity and energy dissipation, especially when using the low-lying antennae and near-ground channels typically found in sensor networks [1]. Additionally, short-range communication enables efficient spatial frequency re-use in sensor networks [1].

Among the three categories (sensing, communication, and data processing) of power consumption, typically, a sensor node spends most of its energy in data communication. This includes both data transmission and reception. Our energy model for the sensors is based on the first order radio model described in [4]. The energy consumed by sensor u to transmit a $k-$bit data packet to sensor v is $T_{uv} = \varepsilon_{elec} \times k + \varepsilon_{amp} \times d_{uv}^2 \times k$, where ε_{elec} is the energy required for transceiver circuitry to process one bit of data, ε_{amp} is the energy required per bit of data for transmitter amplifier, and d_{uv} is the distance between u and v. Transmitter amplifier is not needed by u to receive data and the energy consumed by u to receive a $k-$bit data packet is $R_u = \varepsilon_{elec} \times k$. Typically, $\varepsilon_{elec} = 50nJ/bit$ and $\varepsilon_{amp} = 0.1nJ/bit/m^2$. This effectively translates to $\varepsilon_{amp} \times d_{uv}^2 \ll \varepsilon_{elec}$, especially when short transmission ranges ($\simeq 1m$) are considered. For the discussion in the rest of this paper, we adopt the approximation that $T_{uv} = R_u$ for $\forall\ u, v \in V$. We further assume that no data aggregation is performed during the transmission of the data.

We consider two classes of data gathering problems: store-and-gather problems and continuous sensing and gathering problems. An energy(power) budget B_u is imposed on each sensor node u. We assume that there is no energy constraint on base station r. We can normalize T_{uv} and R_u to 1. For the store-and-gather problems, B_u represents the total number of data packets that u can send and receive. For the continuous sensing and gathering problems, B_u represents the total number of data packets that u can send and receive in one unit of time. To simplify our discussions, we ignore the energy consumption of the sensors when sensing the environment. However, the rate at which sensor $u \in V_c$ can collect data from the environment is limited by the maximum sensing frequency g_u.

Communication link (u, v) has transmission bandwidth c_{uv}. We do not require the communication links to be identical. Two communication links may have different transmission latencies and/or bandwidth. Symmetry is not required either. It may be the case that $c_{uv} \neq c_{uv}$. If $(u, v) \notin E$, then we define $c_{uv} = 0$.

2.2 Store-and-Gather Problems

In store-and-gather problems, the information from the environment is sensed (possibly during a prolonged period) and stored locally at the sensors. These locally stored data are then transferred to the base station during the data gathering stage. This represents those data-oriented applications (e.g. counting the occurrences of endangered birds in a particular region) where the environment changes slowly. There is typically no deadline (or the deadline is loose enough to be ignored) on the duration of data gathering for such problems, and we are not interested in the speed at which the data is gathered. But due to the energy constraint, not all the stored data can be gathered by the base station, and we want to maximize the amount of data gathered.

For each $u \in V_c$, we assume that u has stored d_u data packet before the data gathering starts. Let $f(u,v)$ represent the number of data packets sent from u to v.

For the simplified scenario where V_c contains a single node s, we have the following problem formulation:

Single Source Maximum Data Volume (SMaxDV) Problem:

Given: A graph $G(V,E)$. Source $s \in V$ and sink $r \in V$. Each node $u \in V - \{r\}$ has energy budget B_u.

Find: A real valued function $f : E \to R$

Maximize: $\sum_{v \in \sigma_s} f(s,v)$

Subject to:

$$\begin{array}{lll} f(u,v) \geq 0 & \text{for } \forall\ (u,v) \in E & (1) \\ \sum_{v \in \sigma_u} f(u,v) + \sum_{v \in \psi_u} f(v,u) \leq B_u & \text{for } u \in V - \{r\} & (2) \\ \sum_{v \in \sigma_u} f(u,v) = \sum_{v \in \psi_u} f(v,u) & \text{for } u \in V - \{s,r\} & (3) \end{array}$$

B_u is the energy budget of u. Since we have normalized both T_{uv} and R_u to 1, the total number of data packets that can be sent and received by u is bounded from above by B_u. Condition 2 above represents the energy constraint of the sensors. Sensors $V - \{s,r\}$ do not generate sensed data, nor should they posses any data packets upon the completion of the data gathering. This is reflected in Condition 3 above. We do not model d_s, the number of data packets stored at s before the data gathering starts. This is because d_s is an obvious upper bound for the SMaxDV problem, and can be handled trivially.

$|V_c| > 1$ represents the general scenario where the event is sensed by multiple sensors. This multi-source data gathering problem is formulated as follows:

Multiple Source Maximum Data Volume (MMaxDV) Problem:

Given: A graph $G(V,E)$. The set of source nodes $V_c \subset V$ and sink $r \in V$. Each node $u \in V - \{r\}$ has energy budget B_u. Each node $v \in V_c$ has d_v data packets that are locally stored before the data gathering starts.

Find: A real valued function $f : E \to R$

Maximize: $\sum_{v \in \sigma_s} f(s,v)$

Subject to:

$$\begin{array}{lll} f(u,v) \geq 0 & \text{for } \forall\ (u,v) \in E & (1) \\ \sum_{v \in \sigma_u} f(u,v) + \sum_{v \in \psi_u} f(v,u) \leq B_u & \text{for } u \in V - \{r\} & (2) \\ \sum_{v \in \sigma_u} f(u,v) = \sum_{v \in \psi_u} f(v,u) & \text{for } u \in V - V_c - \{r\} & (3) \\ \sum_{v \in \sigma_u} f(u,v) \leq \sum_{v \in \psi_u} f(v,u) + d_u & \text{for } u \in S_c & (4) \end{array}$$

Similar to the SMaxDV problem, the net flow out of the intermediate nodes ($V - V_c - \{r\}$) is 0 in the MMaxDV problem, as is specified in Condition 3. For each source node $u \in V_c$, the net flow out of u cannot exceed the number of data packets previously stored at u. This is specified in Condition 4.

2.3 Continuous Sensing and Gathering Problems

The continuous sensing and gathering problems model those time critical applications that need to gather as much information as possible from the environment while the nodes are sensing. Examples of such applications include battle field surveillance, target tracking, etc. We want to maximize the total number of data packets that can be gathered by the base station r in one unit of time. We assume that the communications are scheduled by time/frequency division multiplexing or channel assignment techniques. We consider the scenario in which B_u is the maximum power consumption rate allowed by u. Let $f(u,v)$ denote the number of data packets sent from u to v in one unit of time.

Similar to the store-and-gather problem, we have the following mathematical formulation when V_c contains a single node s.

Single Source Maximum Data Throughput (SMaxDT) Problem:

Given: A graph $G(V,E)$. Source $s \in V$ and sink $r \in V$. Each node $u \in V - \{r\}$ has energy budget B_u. Each edge $(u,v) \in E$ has capacity c_{uv}.

Find: A real valued function $f : E \rightarrow R$

Maximize: $\sum_{v\in\sigma_s} f(s,v)$

Subject to:

$0 \le f(u,v) \le c_{uv}$ for $\forall\ (u,v) \in E$ (1)

$\sum_{v\in\sigma_u} f(u,v) + \sum_{v\in\psi_u} f(v,u) \le B_u$ for $u \in V - \{r\}$ (2)

$\sum_{v\in\sigma_u} f(u,v) = \sum_{v\in\psi_u} f(v,u)$ for $u \in V - \{s,r\}$ (3)

The major difference between the SMaxDV and the SMaxDT problem is the consideration of link capacities. In the SMaxDV problem, since there is no deadline for the data gathering, the primary factor that affects the maximum number of gathered data is the energy budgets of the sensors. But for the SMaxDT problem, the number of data packets that can be transferred over a link in one unit of time is not only affected by the energy budget, but also bounded from above by the capacity of that link, as is specified in Condition 1 above. For the SMaxDT problem, we do not model the impact of g_u because g_u is an obvious upper bound of the throughput and can be handled trivially.

Similarly, we can formulate the multiple source maximum data throughput problem as follows.

Multiple Source Maximum Data Throughput (MMaxDT) Problem:

Given: A graph $G(V,E)$. The set of source nodes $V_c \subset V$ and sink $r \in V$. Each node $u \in V - \{r\}$ has energy budget B_u. Each edge $(u,v) \in E$ has capacity c_{uv}.

Find: A real valued function $f : E \rightarrow R$

Maximize: $\sum_{v\in\sigma_s} f(s,v)$

Subject to:

$0 \le f(u,v) \le c_{uv}$ for $\forall\ (u,v) \in E$ (1)

$\sum_{v\in\sigma_u} f(u,v) + \sum_{v\in\psi_u} f(v,u) \le B_u$ for $u \in V - \{r\}$ (2)

$\sum_{v\in\sigma_u} f(u,v) = \sum_{v\in\psi_u} f(v,u)$ for $u \in V - V_c - \{r\}$ (3)

$\sum_{v\in\sigma_u} f(u,v) \le \sum_{v\in\psi_u} f(v,u) + g_u$ for $u \in V_c$ (4)

Condition 4 in the above problem formulation reflects the maximum sensing frequency of the sensors. By associating a certain amount of energy dissipation with each data packet sensed from the environment, we can even model the energy consumption in sensing the environment. Actually, our algorithm (discussed next) is capable of handling this scenario with a few minor modifications. The detailed discussion is omitted here due to space limitations.

3 Flow Maximization with Constraint on Vertices

3.1 Problem Reductions

In this section, we present the formulation of the constrained flow maximization problem where the vertices have limited capacities (*CFM* problem). The CFM problem is an abstraction of the four problems discussed in Section 2.

In the CFM problem, we are given a directed graph $G(V, E)$ with vertex set V and edge set E. Vertex u has capacity constraint $w_u > 0$. Edge (u, v) starts from vertex u, ends at vertex v, and has capacity constraint $c_{uv} > 0$. If $(u, v) \notin E$, we define $c_{uv} = 0$. We distinguish two vertices in G, source s, and sink r. A flow in G is a real valued function $f : E \rightarrow R$ that satisfies the following constraints:

1. $0 \leq f(u,v) \leq c_{uv}$ for $\forall\ (u,v) \in E$. This is the capacity constraint on edge (u, v).
2. $\sum_{v\in\sigma_u} f(u,v) = \sum_{v\in\psi_u} f(v,u)$ for $\forall\ u \in V - \{s,r\}$. This represents the flow conservation. The net amount of flow that goes through any of the vertices, except s and t, is zero.
3. $\sum_{v\in\sigma_u} f(u,v) + \sum_{v\in\psi_u} f(v,u) \leq w_u$ for $\forall\ u \in V$. This is the capacity constraint of vertex u. The total amount of flow going through u cannot exceed w_u. This condition differentiates the CFM problem from the standard network flow problem.

The *value* of a flow f, denoted as $|f|$, is defined as $|f| = \sum_{v\in\sigma_s} f(s,v)$, which is the net flow that leaves s. In the CFM problem, we are given a graph with vertex and edge constraint, a source s, and a sink r, and we wish to find a flow with the maximum value.

It is straight forward to show that the SMaxDV and the SMaxDT problems reduce to the CFM problem. By adding a hypothetical super source node, the MMaxDV and the MMaxDT problems can also be reduced to SMaxDV and SMaxDT, respectively.

The CFM problem has been studied before [7]. Our focus in this paper is to design a distributed and adaptive algorithm for the proposed problem. It can be shown that the CFM problem reduces to a standard network flow problem. Due to the existence of condition 1, condition 3 is equivalent to $\sum_{v\in\sigma_u} f(u,v) \leq w_u/2$ for $u \in V - \{s,r\}$. This means that the total amount of flow out of vertex u cannot exceed $w_u/2$. Suppose we split u ($u \in V - \{s,r\}$) into two nodes u_1 and u_2, re-direct all incoming links to u to arrive at u_1 and all the outgoing links from u to leave from u_2, and add a link from u_1 to u_2 with capacity $w_u/2$, then the vertex constraint w_u is fully represented by the capacity of link (u_1, u_2). Actually, such a split transforms all the vertex constraints to the corresponding link capacities, and effectively reduces the CFM problem to a standard network flow problem. A similar reduction can be found in [7].

The standard network flow problem is stated below:
Given: graph $G(V, E)$, source node $s \in V$, and sink node $r \in V$. Link (u, v) has capacity c_{uv}.
Maximize: $\sum_{v\in\sigma_s} f(s,v)$
Subject to:

$$0 \leq f(u,v) \leq c_{uv} \qquad \text{for } \forall\ (u,v) \in E \qquad (1)$$
$$\sum_{v\in\sigma_u} f(u,v) = \sum_{v\in\psi_u} f(v,u) \qquad \text{for } u \in V - \{s,r\} \qquad (2)$$

3.2 Relationship to Sensor Network Scenarios

The vertex capacity w_u in the CFM problem models the energy(power) budget B_u of the sensor nodes. B_u does not have to be the total remaining energy of u. For example, when the remaining battery power of a sensor is lower than a particular level, the sensor may limit its contribution to the data gathering operation by setting a small value for B_u (so that this sensor still has enough energy for future operations). For another example, if a sensor is deployed in a critical location so that it is utilized as a gateway to relay data packets to a group of sensors, then it may limit its energy budget for a particular data gathering operation, thereby conserving energy for future operations. These considerations can be captured by vertex capacity w_u in the CFM problem.

The edge capacity in the CFM problem models the communication rate (meaningful for continuous sensing and gathering problems) between adjacent sensor nodes. The edge capacity captures the available communication bandwidth between two nodes, which may be less than the the maximum available rate. For example, a node may reduce its radio transmission power to save energy, resulting in a less than maximum communication rate. This capacity can also vary over time based on environmental conditions. Our decentralized protocol results in an on-line algorithm for this scenario.

Because energy efficiency is a key consideration in WSNs, various techniques have been proposed to explore the trade-offs between processing/communication speed and energy consumption. This results in the continuous variation of the performance of the nodes. For example, the processing capabilities may change as a result of dynamic voltage scaling [8]. The data communication rate may change as a result of modulation scaling [11]. As proposed by various studies on energy efficiency, it is necessary for sensors to maintain a power management scheme, which continuously monitors and adjusts the energy consumption and hence changes the computation and communication performance of the sensors. In data gathering problems, these energy related adjustments translate to changes of parameters (node/link capacities) in the problem formulations. Determining the exact reasons and mechanisms behind such changes is beyond the scope of this paper. Instead, we focus on the development of data gathering algorithms that can adapt to such changes.

4 Distributed and Adaptive Algorithm To Maximize Flow

In this section, we first show that the maximum flow remains the same even if we relax the flow conservation constraint. Then we develop a distributed and adaptive algorithm based on this relaxation.

4.1 Relaxed Flow Maximization Problem

Consider the simple example in Figure 1 where s is the source, r is the sink, and u is the only intermediate node. Obviously, the flow is maximized when $f(s,u) = f(u,r) = 10$. Suppose s, u, and r form an actual system and s has sent 10 data packets to u. Then u can send no more than 10 data packets to r even if we permit u to transfer more to r, because u has received only 10 data packets. This means the actual system still works as if $f(u,r) = 10$ even if we set $f(u,r) \geq 10$.

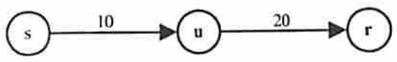


Figure 1. An example of the relaxed network flow problem where $c_{su} = 10$ and $c_{ur} = 20$.

This leads to the following relaxed network flow problem:
Given: graph $G(V, E)$, source node $s \in V$, and sink node $r \in V$. Link (u, v) has capacity c_{uv}.
Maximize: $|f| \overset{\text{def}}{=} \sum_{v \in \sigma_s} f(s, v)$
Subject to:

$$
\begin{array}{lll}
0 \leq f(u, v) \leq c_{uv} & \text{for } \forall \, (u, v) \in E & (1) \\
\sum_{v \in \sigma_u} f(u, v) \geq \sum_{v \in \psi_u} f(v, u) & \text{for } u \in V - \{s, r\} & (2)
\end{array}
$$

Condition 2 differentiates the relaxed and the standard network flow problem. In the relaxed problem, the total flow out of a node is larger than or equal to the total flow into the node, and flow conservation is not required. A feasible function f (which satisfies the two constraints above) to the relaxed flow problem is called a *relaxed flow* in graph G. $|f|$ denotes the net amount of flow out of source s and is called the *value* of the realxed flow. The following theorem shows the relation between the relaxed and the standard network flow problem.

Theorem 1. *Given graph $G(V, E)$, source s and sink r. If f^* is an optimal solution to the relaxed network flow problem, then there exists an optimal solution f' to the standard network flow problem such that $f'(u, v) \leq f^*(u, v)$ for $\forall \, (u, v) \in E$. Additionally, $|f^*| = |f'|$.*

Proof of the theorem is not difficult and omitted here due to space limitations. If we interpret $f^*(u, v)$ as the number of data units that we ask u to transfer and $f'(u, v)$ as the number of data units that u actually transfers, then this theorem essentially indicates that the solution to a relaxed flow problem can have an actual implementation that satisfies flow conservation!

4.2 The Algorithm

In this section, we develop a decentralized and adaptive algorithm for the relaxed network flow maximization problem. This algorithm is a modified version of the Push-Relabel algorithm [2] and is denoted as the *Relaxed Incremental Push-Relabel* (RIPR) algorithm.

The Push-Relabel algorithm is a well known algorithm for network flow maximization. It has a decentralized implementation where every node only needs to exchange messages with its immediate neighbors and makes decisions locally. But in order to be adaptive to the changes in the system, this algorithm has to be re-initialized and re-run from scratch each time when some parameters (weight of the nodes and edges in the graph) of the flow maximization problem change. Each time before starting to search for the new optimal solution, the algorithm needs to make sure that every node has finished its local initialization, which requires a global synchronization and compromises the property of decentralization.

In contrast to the Push-Relabel algorithm, our algorithm introduces the adaptation operation, which is performed upon the current values of $f(u, v)$ and $h(u)$ for $\forall u, v \in V$. In other words, our algorithm performs *incremental* optimization as the parameters of the system change. Our algorithm does not need global synchronizations. Another difference is that our algorithm applies to the relaxed network flow problem, rather than the standard one.

For the discussion below, let us first briefly re-state some notations for the network flow maximization problem. For notational convenience, if edge $(u, v) \notin E$, we define $c_{uv} = 0$; if the actual data transfer is from u to v, we define $f(v, u) = -f(u, v)$. With these two definitions, if neither (u, v) nor (v, u) belongs to E, then $c_{uv} = c_{vu} = 0$, which implies that $f(u, v) = f(v, u) = 0$. Of course, $f(u, u) = -f(u, u)$

implies that $f(u,u) = 0$, which essentially says that a node cannot send flow to itself. In this way, we can define $f(u,v)$ over $V \times V$, rather than being restricted to E. $f(u,v) = -f(v,u)$ also allows us to compute the total amount of flow into u as $\sum_{v \in \psi_u \cup \sigma_u} f(v,u)$, which equals $\sum_{v \in V} f(v,u)$ since $f(v,u) = f(u,v) = 0$ if $(u,v) \notin E$ and $(v,u) \notin E$. With the definition of $f(u,v)$ thus extended, it is easy to show that the relaxed network flow problem is equivalent to the following formulation:

Given: graph $G(V,E)$, source node $s \in V$, and sink node $r \in V$. Link (u,v) has capacity c_{uv}. $c_{uv} = 0$ if $(u,v) \notin E$.

Maximize: $|f| \overset{\text{def}}{=} \sum_{v \in V} f(s,v)$

Subject to:

$$\begin{array}{lll} f(u,v) = -f(v,u) & \text{for } u,v \in V & (1) \\ f(u,v) \le c_{uv} & \text{for } u,v \in V & (2) \\ \sum_{v \in V} f(v,u) \le 0 & \text{for } u \in V - \{s,r\} & (3) \end{array}$$

Given a direct graph $G(V,E)$, function f is called a flow if it satisfies the three conditioins in the above problem; function f is called a pre-flow if it satisfies conditions 1 and 2. Given $G(V,E)$ and f, the *residual capacity* $c_f(u,v)$ is given by $c_{uv} - f(u,v)$, and the *residual network* of G induced by f is $G_f(V,E_f)$, where $E_f = \{(u,v) | u \in V, v \in V, c_f(u,v) > 0\}$. For each node $u \in V$, $e(u)$ is defined as $e(u) = \sum_{v \in V} f(v,u)$, which is the total amount of flow into u.

The algorithm is as follows:

1. *Initialization:* $h(u)$, and $f(u,v)$ are initialized as follows:

$$\begin{array}{lll} h(u) & = 0 & \text{for } \forall\, u \in V \\ f(u,v) & = 0 & \text{for } \forall\, u,v \in E \\ h(s) & = |V| & \\ f(s,v) & = l_{sv} & \text{for } \forall\, v \in V \\ f(v,s) & = -l_{vs} & \text{for } \forall\, v \in V \\ e(u) & = \sum_{v \in V} f(v,u) & \text{for } \forall\, u \in V \end{array}$$

2. *Search for maximum flow:*

   Each node $u \in V - \{s,r\}$ conducts one of the following three operations as long as $e(u) \neq 0$:

   (a) $Push(u,v)$: applies when $e(u) > 0$ and $\exists (u,v) \in E_f$ s.t. $h(u) > h(v)$,

$$\begin{array}{ll} d & = \min(e(u), c_f(u,v)) \\ f(u,v) & = f(u,v) + d \\ f(v,u) & = -f(u,v) \\ e(u) & = e(u) - d \\ e(v) & = e(v) + d \end{array}$$

   (b) $Relabel(u)$: applies when $e(u) > 0$ and $h(u) \le h(v)$ for $\forall\,(u,v) \in E_f$,

$$h(u) = \min_{(u,v) \in E_f} h(v) + 1$$

3. *Adaptation to changes in the system:* For the flow maximization problem, the only possible change that can occur in the system is the increase or decrease of the capacity of some edges. Suppose the value of l_{uv} changes to l'_{uv}, the following four scenarios are considered when performing the Adaptation (u, v) operation:

   (a) if $l'_{uv} > l_{uv}$ and $f(u, v) < l_{uv}$, do nothing.
   (b) if $l'_{uv} > l_{uv}$ and $f(u, v) = l_{uv}$, then

$$\begin{array}{lll} h(s) & = h(s) + 3|V| & \\ f(s,v) & = l_{sv} & \text{for } \forall\, v \in \sigma_s \\ f(v,s) & = -l_{sv} & \text{for } \forall\, v \in \sigma_s \\ e(v) & = \sum_{v \in V} f(v,v) & \text{for } \forall\, v \in V \end{array}$$

   (c) if $l'_{uv} < l_{uv}$ and $f(u, v) \leq l'_{uv}$, do nothing.
   (d) if $l'_{uv} < l_{uv}$ and $f(u, v) > l'_{uv}$, then

$$\begin{array}{lll} h(s) & = h(s) + 3|V| & \\ f(s,v) & = l_{sv} & \text{for } \forall\, v \in \sigma_s \\ f(v,s) & = -l_{sv} & \text{for } \forall\, v \in \sigma_s \\ e(v) & = \sum_{v \in V} f(v,v) & \text{for } \forall\, v \in V \\ f(u,v) & = l'_{uv} & \\ f(v,u) & = -l'_{uv} & \\ e(u) & = e(u) + (l_{uv} - l'_{uv}) & \\ e(v) & = e(v) - (l_{uv} - l'_{uv}) & \end{array}$$

The above algorithm defines an integer valued auxiliary function $h(u)$ for $u \in V$, which will be discussed below. The 'adaptation' is activated when some link capacity changes in the relaxed flow problem. Because link capacities in the relaxed flow problem map to either vertex or link capacities in the corresponding CFM problem, the adaptation operation actually reacts to capacity changes in both vertex and link capacities. The 'Push' and 'Relabel' operations are called the *basic operations*. Every node in the graph determines its own behavior based on the knowledge about itself and its neighbors (as can be seen, the Push and Relabel operations are triggered by variables whose value are locally known by the nodes). No central coordinator or global information about the system is needed. More importantly, unlike the Push-Relabel algorithm, no global synchronization is needed when the RIPR algorithm adapts to the changes in the system.

An intuitive explanation of the RIPR is as follows. $h(u)$ represents, intuitively, the shortest distance from u to t when $h(u) \leq h(s)$. When $h(u) > h(s)$, $h(u) - h(s)$ represents the shortest distance from u to s. Hence the RIPR algorithm attempts to push more flow from s to t along the shortest path; excessive flow of intermediate nodes are pushed back to s along the shortest path. Similar to the Edmonds-Karp algorithm[2], such a choice of paths can lead to an optimal solution.

Lemma 1. *During the execution of the RIPR Algorithm, for $\forall u \in V$, $h(u)$ never decreases.*

Proof: $h(s)$ is only changed by the Adaptation operation, during which $h(s)$ is increased by $2|V|$. $h(r)$ never changes. When $u \neq s$ and $u \neq r$, $h(u)$ is only changed by $Relabel(u)$. $Relabel(u)$ is applied when $e(u) > 0$ and $h(u) \leq h(v)$ for $\forall\, v \in \{v | c_f(u,v) > 0\}$. And $h(u_i) = \min_{(u,v) \in E_f} h(v) + 1$ after $Relabel(u)$. Hence $h(u)$ is increased at least by 1. □

Lemma 2. *During the execution of the RIPR Algorithm, for $\forall u \in V$ s.t. $e(u) > 0$, there exists a simple path in E_f from u to a node v s.t. $e(v) < 0$.*

Proof: Suppose $e(u) > 0$. Let $\hat{V}$ denotes the set of nodes that can be reached by u through a simple path in E_f. Note that $u \in \hat{V}$. For sake of condiction, suppose $e(v) \geq 0$ for $\forall\, v \in \hat{V}$. Let $\overline{V} = V - \hat{V}$.

We claim that if $x \in \overline{V}$ and $y \in \hat{V}$, then $f(x,y) \leq 0$. Otherwise if $f(x,y) > 0$, then $c_f(y,x) = c_{yx} - f(y,x) = c_{yx} + f(x,y) > 0$, which means x can be reached from y in E_f, hence there exists a path from u to x in E_f. But this contradicts the choice that $x \in \overline{V}$.

It is fairly easy to show that $\sum_{v\in\hat{V}} e(v) = \sum_{x\in\overline{V},\, y\in\hat{V}} f(x,y)$. Hence $\sum_{v\in\hat{V}} e(v) \leq 0$. But this contradicts the assumption that $e(v) \geq 0$ for $\forall\, v \in \hat{V}$ and $e(u) > 0$. □

Lemma 3. *During the execution of the RIPR, for $\forall\, u \in V$, if $e(u) > 0$, then either $Relabel(u)$, or $Push(u,v)$ (where $v \in V$) can be applied.*

Proof: When $e(u) > 0$, if $\exists v$ s.t. $c_f(u,v) > 0$ and $h(u) > h(v)$, then $Push(u,v)$ can be applied; otherwise, $h(u) \leq h(v)$ for $\forall\, v \in \{v | c_f(u,v) > 0\}$, which means $Relabel(u)$ can be applied. □

Lemma 4. *During the execution of the RIPR algorithm,*

$$\begin{array}{ll} h(u) \leq \max(h(v)+1, h(s)-|V|-1) & \textit{for } \forall\,(u,v) \in E_f, \\ h(u) \leq h(s)+|V|-1 & \textit{for } \forall\, u \in V \end{array}$$

Proof: We prove by induction on the number of adaptation operations.

- Base case:

  Before any changes occur in the system, the adaptation operation will not be applied. At this stage, the Incremental Push-Relabel algorithm performs the exact operations as the Push-Relabel algorithm, hence we have

  $$\begin{array}{ll} h(u) \leq \max(h(v)+1, h(s)-|V|-1) & \text{for } \forall\,(u,v) \in E_f, \\ h(u) \leq h(s)+|V|-1 & \text{for } \forall\, u \in V \end{array}$$

  before any adaptation operation is applied.

- Induction step:

  Suppose after the adaptation has been applied $n-1$ times and we still have

  $$\begin{array}{ll} h(u) \leq \max(h(v)+1, h(s)-|V|-1) & \text{for } \forall\,(u,v) \in E_f, \\ h(u) \leq h(s)+|V|-1 & \text{for } \forall\, u \in V \end{array}$$

  and then the n^{th} adaptation, $Adaptation(u^*, v^*)$, is applied.

  1. We first show that $h(u) \leq \max(h(v)+1, h(s)-|V|-1)$ for $\forall\,(u,v) \in E_f$ after $Adaptation(u^*, v^*)$. Considering $Adaptation(u^*, v^*)$, if either scenario (a) or (c) occurs, no residual edge is added or removed, no node $w \in V$ has its $h(w)$ changed, either. If scenario (d) occurs, the change in the system removes (u^*, v^*) from E_f and hence the corresponding constraint on $h(u^*)$

and $h(v^*)$. If scenario (b) occurs, (u^*, v^*) is added to the E_f. By induction assumption, $h(u^*) \leq h(s) + |V| - 1$ before the adaptation operation. Because $h(u^*)$ does not change and $h(s)$ increases by $2|V|$ after the operation, $h(u^*) \leq h(s) - |V| - 1$ after the adaption operation. In summary, after the adaptation operation, $h(u) \leq \max(h(v)+1, h(s)-|V|-1)$ for $\forall\ (u, v) \in E_f$.

Adaptation(u^*, v^*) changes the values of some $e(w)$, allowing new Push and Relabel operations to be applied. Yet these operations preserve the property that $h(u) \leq \max(h(v) + 1, h(s) - |V| - 1)$ for $\forall\ (u, v) \in E_f$. This is shown by induction on the number of Push and Relabel operations.

(a) Suppose*Push*(u, v) is applied.
This may add edge (v, u) into E_f or remove the edge (u, v) from E_f. In the former case, for edge $(v, u) \in E_f$, we have $h(v) < h(u)$ because otherwise the push will not be applied. In the latter case, the removal of (u, v) from E_f removes the corresponding constraint on $h(u)$ and $h(v)$. In both cases, we still have $h(u) \leq \max(h(v) + 1, h(s) - |V| - 1)$ for any $(u, v) \in E_f$.

(b) Suppose *Relabel*(u) is applied.
For a residual edge (u, v) that leaves u, we have $h(u) = \min_{w \in \{w|(u,w) \in E_f\}} h(w) + 1$ after the Relabel operation, which means $h(u) \leq h(v) + 1$. For a residual edge (w, u) that enters u, $h(w) \leq \max(h(u) + 1, h(s) - |V| - 1)$ before the relabel operation. According to Lemma 1, $h(w) \leq \max(h(u)+1, h(s)-|V|-1)$ after the relabel operation. Therefore, after a relabel operation, we have $h(u) \leq \max(h(v) + 1, h(s) - |V| - 1)$ for any $(u, v) \in E_f$.

2. Now we need to show that $h(u) \leq h(s) + |V| - 1$ for $\forall\ u \in V$.
Let $\hat{V}$ denote the set of $u \in V$ s.t. there exists a simple path from u to s in E_f. $\overline{V} = V - \hat{V}$.
For any node $u \in \hat{V}$, suppose the simple path to s in E_f is $\{u, u_1, ..., u_k\}$, where $u_k = s$ and $k \leq |V| - 1$. We have

$$\begin{aligned} h(u) &\leq \max(h(u_1) + 1, h(s) - |V| - 1) \\ h(u_1) &\leq \max(h(u_2) + 1, h(s) - |V| - 1) \\ &\dots \\ h(u_{k-1}) &\leq \max(h(u_k) + 1, h(s) - |V| - 1) \end{aligned}$$

Combining these inequalities, we have

$$\begin{aligned} h(u) &\leq \max(h(u_k) + k, h(s) - |V| - 1 + k - 1) \\ &= \max(h(s) + k, h(s) - |V| + k - 2) \\ &\leq \max(h(s) + |V| - 1, h(s) - 3) \\ &= h(s) + |V| - 1 \end{aligned}$$

For any node $u \in \overline{V}$, according to Lemma 2, there exist a simple path in E_f from u to a node z s.t. $e(z) < 0$ and $z \neq s$. Suppose the simple path is $\{u, u_1, ..., u_k\}$, where $u_k = z$ and $k \leq |V| - 1$. We have

$$\begin{aligned} h(u) &\leq \max(h(u_1) + 1, h(s) - |V| - 1) \\ h(u_1) &\leq \max(h(u_2) + 1, h(s) - |V| - 1) \end{aligned}$$

$$\dots$$

$$h(u_{k-1}) \leq \max(h(u_k) + 1, h(s) - |V| - 1)$$

Note that $e(u) \geq 0$ immediately after the initialization for $\forall\ u \in V$. The only operation that can bring $e(z)$ below 0 is the adaptation operation (when scenario (d) occurs). Suppose $e(z)$ becomes negative as the result of m^{th} adaptation operation ($m \leq n$). Since the Relabel operation (which is the only operation that can increase the value of $h(z)$) is applied only if $e(z) > 0$, then $e(z) < 0$ means that $h(z)$ has not been increased after the m^{th}, and hence the n^{th} adaptation operation. Therefore, $h(z) \leq h(s) + |V| - 1$ before the n^{th} adaptation means that $h(z) \leq h(s) - |V| - 1$ thereafter, since $h(s)$ is increased by $2|V|$.

Combining these inequalities, we have

$$\begin{aligned} h(u) &\leq \max(h(V_{i_k}) + k, h(s) - |V| - 1 + k - 1) \\ &= \max(h(V_s) + k, h(s) - |V| - 2 + k) \\ &\leq \max(h(V_0) - |V| - 1 + |V| - 1, h(s) - |V| - 2 + |V| - 1) \\ &\leq h(V_0) + |V| - 1 \end{aligned}$$

□

Corollary 1. *During the execution of the RIPR algorithm, for any node $u \in V$, if $e(u) < 0$, then $h(u) \leq h(s) - |V| - 1$.*

The proof of Corollary 1 is included in the proof of Lemma 4.

Lemma 5. *During the execution of the RIPR algorithm, for $\forall\ u \in V - \{s, r\}$, if there exists a simple path from s to u in E_f, then $e(u) \geq 0$.*

Proof: Suppose for the sake of contradiction that there exists a node $u \in V - \{s, r\}$ such that $e(u) < 0$ and there exists a simple path $\{s, u_1, ..., u_k, u\}$ in E_f. Without loss of generality, this is a simple path and $k \leq |V| - 2$.

According to Lemma 4,

$$\begin{aligned} h(u_1) &\leq \max(h(u_2) + 1, h(s) - |V| - 1) \\ &\dots \\ h(u_{k-1}) &\leq \max(h(u_k) + 1, h(s) - |V| - 1) \\ h(u_k) &\leq \max(h(u) + 1, h(s) - |V| - 1) \end{aligned}$$

According to Corollary 1, $h(u) \leq h(s) - |V| - 1$ since $e(u) < 0$. Combining these inequalities, we can see that

$$\begin{aligned} h(u_1) &\leq \max(h(u) + k, h(s) - |V| - 2 + k) \\ &\leq \max(h(s) - |V| - 1 + k, h(s) - |V| - 2 + k) \\ &\leq h(s) - |V| - 1 + |V| - 2 \\ &< h(s) \end{aligned}$$

On the other hand, consider the first hop (s, u_1) along this path. $(s, u_1) \in E_f$ implies that $f(s, u_1) < c_{su_1}$. Recall that the value of $f(s, u_1)$ is set to c_{su_1} after the initialization and each adaptation operation.

The only operation that can reduce the value of $f(s, u_1)$ is a push from u_1 to s. However, $Push(u_1, s)$ is applied only when $h(u_1) > h(s)$. This contradicts the claim $h(u_1) < h(s)$ that we just derived.

$\square$

Similar to the standard flow problem, for the relaxed flow problem, a *cut* is defined as a binary partition (S, R) of V such that $S \cup R = V$, $S \cap R = \emptyset$, $s \in S$ and $r \in R$. The capacity c_{SR} of a cut (S, R) is defined as $c_{SR} = \sum_{u \in S, v \in R} c_{uv}$. The next lemma shows that the value of a relaxed flow cannot exceed the capacity of any cuts.

Lemma 6. *Given graph $G(V, E)$ with source s and sink r, a relaxed flow f, and an arbitrary cut (S, R) of G, $\sum_{v \in V} f(s, v) \leq c_{SR}$.*

Proof: we have $e(u) \leq 0$ for $u \in V - \{s, r\}$. Therefore

$$\begin{aligned}
&\textstyle\sum_{u \in S-\{s\}} e(u) = \sum_{v \in V, u \in S-\{s\}} f(v, u) \leq 0 \\
&\Rightarrow \textstyle\sum_{v \in S, u \in S-\{s\}} f(v, u) + \sum_{v \in R, u \in S-\{s\}} f(v, u) \leq 0 \\
&\Rightarrow \textstyle\sum_{u \in S-\{s\}} f(s, u) + \sum_{v \in S-\{s\}, u \in S-\{s\}} f(v, u) + \sum_{v \in R, u \in S-\{s\}} f(v, u) \leq 0 \\
&\Rightarrow \textstyle\sum_{u \in S-\{s\}} f(s, u) + \sum_{v \in S-\{s\}, u \in S-\{s\}} f(v, u) \leq \sum_{v \in R, u \in S-\{s\}} f(u, v) \\
&\Rightarrow \textstyle\sum_{u \in S-\{s\}} f(s, u) \leq \sum_{v \in R, u \in S-\{s\}} f(u, v) \\
&\Rightarrow \textstyle\sum_{u \in S-\{s\}} f(s, u) + \sum_{u \in R} f(s, u) \leq \sum_{v \in R, u \in S-\{s\}} f(u, v) + \sum_{u \in R} f(s, u) \\
&\Rightarrow \textstyle\sum_{u \in V-\{s\}} f(s, u) \leq \sum_{u \in S, v \in R} f(u, v)
\end{aligned}$$

Since $f(u, v) \leq c_{uv}$ for $\forall\, u, v \in V$, additionally, because $f(s, s) = 0$, we have

$$\textstyle\sum_{v \in V} f(s, v) = \sum_{u \in V-\{s\}} f(s, u) \leq \sum_{u \in S, v \in R} c_{uv} = c_{SR}$$

$\square$

Lemma 7. *If the RIPR algorithm terminates, it finds the maximum relaxed flow.*

Proof:

According to Lemma 3, if the algorithm terminates, then $e(u) \leq 0$ for $\forall\, u \in V - \{s, r\}$. Hence f is a flow if the algorithm terminates.

Given such an f, we construct a cut of G as follows:

$$S = \{u \in V | \text{there exits a simple path from } s \text{ to } u \text{ in } E_f\}$$

$$R = V - S$$

According to Lemma 5, $e(u) \geq 0$ for $u \in S - \{s\}$. Note that $e(u) \leq 0$ for $\forall\, u \in V - \{s, r\}$ upon teriminiation of the algorithm. Hence $e(u) = 0$ (i.e. $\sum_{v \in V} f(v, u) = 0$) for $\forall\, u \in S - \{s\}$. Then it is easy to show that $\sum_{u \in S} f(s, u) = \sum_{u \in S-\{s\}} f(s, u) = \sum_{u \in S-\{s\}, v \in R} f(u, v)$.

Therefore,

$$\begin{aligned}
\textstyle\sum_{v \in V} f(s, v) &= \textstyle\sum_{v \in S} f(s, v) + \sum_{v \in R} f(s, v) \\
&= \textstyle\sum_{u \in S-\{s\}, v \in R} f(u, v) + \sum_{v \in R} f(s, v) \\
&= \textstyle\sum_{u \in S, v \in R} f(u, v)
\end{aligned}$$

We claim that $f(u,v) = c_{uv}$ for $\forall\, u \in s, v \in R$, because otherwise $f(u,v) \leq c_{uv}$ implies that edge $(u,v) \in E_f$, hence v can be reached by s in E_f. But this contradicts the definition of R.

Therefore,

$$\textstyle\sum_{v \in V} f(s,v) = \sum_{u \in S, v \in R} c(u,v) = c_{SR}$$

Accordign to Lemma 6, such a relaxed flow f is a maximum relaxed flow. $\square$

Lemma 8. *If the adaptation is applied n $(n \geq 0)$ times, then the number of Relabel operations that can be performed is less than* $((2n+2)|V|^2$.

Proof: If the adaptation is applied n times, then $h(s) = (2n+1)|V|$. According to Lemma 4, $h(u) \leq h(s) + |V| - 1 = (2n+2)|V| - 1$ for $\forall\, u \in V$. Each time $Relabel(u)$ is applied, $h(s)$ is increased at least by 1. Since $h(s) = 0$ initially, $Relabel(u)$ is applied at most $(2n+2)|V| - 1$ times. There are $|V|$ nodes in the system, hence then the total number of Relabel operations that can be performed is at most $((2n+2)|V| - 1)|V|$, which is less than $(2n+2)|V|^2$. $\square$

If a $Push(u,v)$ operation removes (u,v) from E_f (i.e. $c_f(u,v) = 0$ after the operation), it is a *saturated push*, otherwise it is a *non-saturated push*.

Lemma 9. *If the adaptation is applied n $(n \geq 0)$ times, then the number of saturated push operations that can be performed is less than* $(n+1)|V| \cdot |E|$.

Proof: Consider edge $(u,v) \in E$. Suppose a saturated push $Push(u,v)$ is first applied. For a second saturated push to be applied over (u,v), $Push(v,u)$ must be applied before the second saturated push. Because $h(u) > h(v)$ for the first push (otherwise the first push will not be applied), then $h(v)$ must be increased at least by 2, otherwise $h(v) \leq h(u)$ then $Push(v,u)$ will not be applied. Similarily, $h(u)$ must be increased at least by 2 for the second saturated push to occur over edge (u,v). So on so force. Because $h(u) \leq h(s) + |V| - 1 = (2n+2)|V| - 1$, $h(u)$ and $h(v)$ cannot increase to infinity. It is easy to show that saturated push can occur at most $((2n+2)|V| - 1)/2$ times for edge (u,v). There are $|E|$ edges in the graph. The total number of saturated push operations is less than $((2n+2)|V| - 1)/2 \cdot |E|$, which is less than $(n+1)|V| \cdot |E|$. $\square$

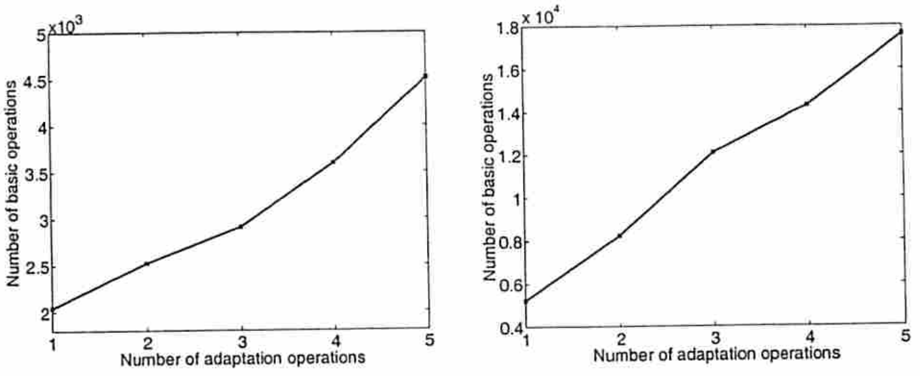
Lemma 10. *If the adaptation is applied n $(n \geq 0)$ times, then the number of non-saturated push operations that can be performed is less than* $(n^2 + 2n + 1) \cdot (4|V|^3 + 2|V|^2|E|)$.

Proof: Define a potential function $\Phi = \sum_{e(u)>0} h(u)$. $\Phi = 0$ initially. Obviously, $\Phi \geq 0$.

According to Lemma 4, $h(u) \leq h(s) + |V| - 1 = (2n+2)|V| - 1$, hence a relabel operation increases Φ by at most $(2n+2)|V| - 1$. According to Lemma 8, there can be at most $(2n+2)|V|^2$ Relabel operations. The increase in Φ induced by all relabel operations is at most $((2n+2)|V| - 1) \cdot (2n+2)|V|^2$. A saturated push $Push(u,v)$ increases Φ by at most $(2n+2)|V| - 1$ since $e(v)$ may become positive after the push, and $(2n+2)|V| - 1$ is the highest value that $h(v)$ can be. According to Lemma 9, the increase in Φ induced by all saturated push is at most $((2n+2)|V| - 1) \cdot ((n+1)|V| \cdot |E|)$.

For a non-saturated push $Push(u,v)$, $e(u) > 0$ before the push and $e(u) = 0$ after the push, hence $h(u)$ is excluded from Φ after the push. If $e(v) > 0$ after the push, Φ is decreased at least by 1 because $h(u) - h(v) > 0$. If $e(v) \leq 0$ after the push, then Φ is decreased by $h(u) \geq 1$.

Therefore, the total increase in Φ is at most $((2n+2)|V| - 1) \cdot (2n+2)|V|^2 + ((2n+2)|V| - 1) \cdot ((n+1)|V| \cdot |E|) < (n^2 + 2n + 1) \cdot (4|V|^3 + 2|V|^2|E|)$, while each non-saturated push decreases Φ at least by 1. Therefore, the total number of non-saturated push operations that can be performed is at most $(n^2 + 2n + 1) \cdot (4|V|^3 + 2|V|^2|E|)$. $\square$



(a) systems with 40nodes and 120 edges

(b) systems with 60 nodes and 180 edges

Figure 2. Complexity of the RIPR algorithm. Number of basic operations vs number of adaptations

Theorem 2. *The RIPR algorithm finds the maximum flow for the relaxed flow problem with $O(n^2 \cdot |V|^2 \cdot |E|)$ basic operations, where n is the number of adaptation operations performed, $|V|$ is the number of nodes in the graph, and $|E|$ is the number of edges in the graph.*

Proof: Immediate from Lemma 7, 8, 9, and 10. □

We have proved above that the RIPR algorithm needs $O(n^2 \cdot |V|^2 \cdot |E|)$ basic operations to find the maximum flow. However, experiments show that the average performance is better than the bound stated in Theorem 2. Figure 2 shows the experimental results. Both Figure 2 (a) and (b) report results averaged over 50 randomly generated systems. The x-axis represents the number of adaptation operations. The y-axis shows the average number of basic operations performed by the algorithm to find the maximum flow. These simulation results suggest that, instead of n^2, there may be a linear relation between the total number of basic operations and the number of adaptations. We are currently investigating the possibility of a better complexity bound.

4.3 A Simple Protocol for Data Gathering

In this section, we show a simple on-line protocol for SMaxDT problem based on the RIPR algorithm in Section 4.

The maximum flow obtained using the RIPR algorithm does not tell us how to transfer the data through the network. It only contains information such as 'u needs to transfer 0.38 unit of data to v in one unit of time'. But the actual system needs to deal with integer number of data packets. Furthermore, before the RIPR algorithm finds the maximum flow, a node, say u, may have a positive valued $e(u)$, which means u is accumulating data at that time instance. Yet such a node u should not keep accumulating data as $e(u)$ will eventually be driven to zero.

These issues are addressed by maintaining a data buffer at each node. Initially, all the data buffers are empty. The source node s senses the environment and fills its buffer continuously. At any time instance, let β_u denote the amount of buffer used by node u. Each node $u \in V$ operates as follows:

1. Contact the adjacent node(s) and execute the RIPR algorithm.

2. While $\beta_u > 0$, send message 'request to send' to all successors v of u s.t. $f(u, v) > 0$. If 'clear to send' is received from v, then set $\beta_u \leftarrow \beta_u - 1$ and send a data packet to v.

3. Upon receiving 'request to send', u acknowledges 'clear to send' if $\beta_u \leq U$. Here U is a pre-set threshold that limits the maximum number of data packets a buffer can hold.

For node s, it stops sensing if $\beta_s > U$. Two types of data are transferred in the system: the control messages that are used by the RIPR algorithm, and the sensed data themselves. The control messages are exchanged among the nodes to query and update the values of $f(u, v)$ and h_u etc. The 'request to send' and 'clear to send' messages are also control messages, though they are more related to data transfer. The control messages and the sensed data are transmitted over the same links and higher priority is given to the control messages in case of a conflict.

For the MMaxDT problem, the situation is a bit more complicated. Since the MMaxDT problem is reduced to the SMaxDT problem by adding a hypothetical super source node s', the RIPR algorithm needs to maintain the flow out of s' as well as the value of function $h(s')$. Additionally, the values of $f(s', v)$ ($v \in V_c$) and $h(s')$ are needed by all nodes $u \in V_c$ during the execution of the algorithm. Because s' is not an actual sensor, sensors in V_c therefore need to maintain a consistent image of s'. This requires some regional coordination among sensors in V_c and may require some extra cost to actually implement such a consistent image.

SMaxDV and MMaxDV are by nature off-line problems and hence we do not develop on-line protocols for them. Actually, we can approximate the optimal solution to SMaxDV and MMaxDV using some simple heuristics. For example, we can find the shortest path (in terms of the number of hops) from the source (or one of the sources) to the sink and push as many data packets as possible along the path until the energy of some node(s) along the path is depleted. Then we find the second shortest path and push data packets along this path. This is repeated till the source and sink are disconnected. Such a heuristic is similar to the augmenting path method for the standard flow problem except that the paths are searched in the original graph, rather than the residual graph. The heuristic may not find the optimal solution. However, it can be shown experimentally that such a heuristic can achieve a close to optimal performance [5].

5 Experimental Results

Simulations were conducted to illustrate the effectiveness of the proposed data gathering algorithm. We focus on the continuous sensing and gathering problems where our simple protocol can be applied.

As we have already proved that the RIPR algorithm finds the maximum flow, we do not show here the throughput achievable by our protocol, which is based on the RIPR algorithm. Actually, throughput up to 95% of the optimal has been observed in the simulations.

As claimed in Section 4, adaptation is an important properties of the proposed algorithm and hence the data gathering protocol. This is illustrated in the following simulations of the SMaxDT problem.

The simulated sensor network was generated by randomly scattering 40 sensor nodes in a unit square. The base station was located at the left-corner of the square. The source node was randomly chosen from the sensor nodes. The number of sensor nodes that can communicate directly to a specific node is determined by a connectivity parameter, $\kappa \in (0, 1]$, such that the average number of neighbors of a sensor node is $200\pi\kappa^2$. B_u's are uniformly distributed between 0 and B_{max}. B_{max} was set to 500 for the simulations. We assume a signal decaying factor of r^{-2}. the flow capacity between sensor nodes u and v

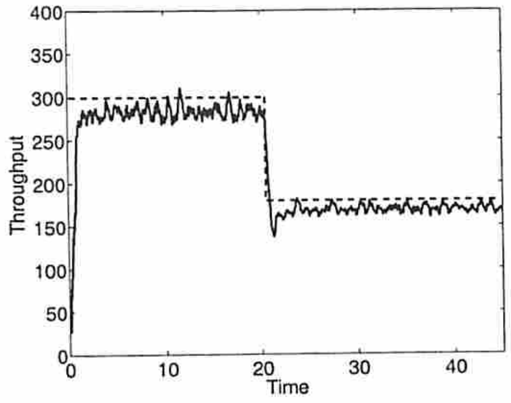


Figure 3. Adaptation to changes in the system.

is determined by Shannon's theorem as $l_{uv} = W \log(1 + \frac{P_{uv} r_{uv}^{-2}}{\eta})$ where W is the bandwidth of the link, r_{uv} is the distance between u and v, P_{uv} is the transmission power on link (u, v), and η is the noise in the communication channel. In all the simulations, W was set to $1KHz$, P_{uv} was set to $10^{-3}mW$, and η was set to $10^{-6}mW$. U was set to 2.

During the course of data gathering, the capacities of a set of randomly selected edges were reduced by 50% at time instance $t = 20s$. The energy budgets of another set of randomly selected sensors were reduced by 30% at $t = 20s$. The achieved data gathering throughput is shown in Figure 3 , where the dotted line represents the optimal throughput calculated offline. Let $N(t)$ represents the total number of data packets received by the base station from time 0 to time instance t. The instantaneous throughput (reported in Figure 3) at time instance τ is calculated as $(N(\tau+0.1)-N(\tau-0.1))/0.2$. Various randomly generated systems were simulated under such settings. The same adaptation trend was observed and only one such result is shown in Figure 3. When the system parameters changed at $t = 20s$, the adaptation procedure was activated and the data gathering was adapted. As can be seen, the system operates at (close to) the new optimal throughput after the adaptation was completed.

6 Conclusion

In this paper, we studied a set of data gathering problems in energy-constrained wireless sensor networks. We reduced such problems to a network flow maximization problem with vertex capacity constraint, which can be further reduced to the standard network flow problem. After deriving a relaxed formulation for the standard network problem, we developed a distributed and adaptive algorithm to maximize the flow. This algorithm can be implemented as a simple data gathering protocol for WSNs.

One of the future directions is to design distributed algorithms that do not generate excessive flow at the nodes (i.e. $e(u)$ does not become positive) during the execution. Our formulation of constrained flow optimizations can be applied to problems beyond those four discussed in this paper. For example, the system model considered in [6] gathers data in rounds. In each round, every sensor generates one data packet and the data packets from all the sensors need to be collected by the sink. The goal is to maximize the total number of rounds the system can operate under energy constraints of the nodes. This problem can be described by our constrained flow formulation and an optimal solution can be developed [5].

Acknowledgment

We thank the anonymous reviewer for pointing out earlier work on constrained flow problems.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cyirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, 2002.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1992.

[3] A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum Flow Problem. *Journal of Association for Computing Machinery*, 35:921–940, 1988.

[4] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy Efficient Communication Protocol for Wireless Micro-sensor Networks. In *Proceedings of IEEE Hawaii International Conference on System Sciences*, 2000.

[5] B. Hong and V. K. Prasanna. Constrained Flow Optimization with Applications to Data Gathering in Sensor Networks. Technical Report CENG-2004-07, Department of Electrical Engineering, University of Southern California, http://www-scf.usc.edu/~bohong/report_may04.ps, May 2004.

[6] K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks. *IEEE Networks '02 Conference*, 2002.

[7] E. L. Lawler. *Combinatorial Optimization : Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[8] R. Min, T. Furrer, and A. Chandrakasan. Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks. *Workshop on VLSI (WVLSI '00)*, April 2000.

[9] F. Ordonez and B. Krishnamachari. Optimal Information Extraction in Energy-Limited Wireless Sensor Networks. *to appear in IEEE Journal on Selected Areas in Communications, special issue on Fundamental Performance Limits of Wireless Sensor Networks, 2004*.

[10] N. Sadagopan and B. Krishnamachari. Maximizing Data Extraction in Energy-Limited Sensor Networks. *IEEE Infocom 2004*, 2004.

[11] C. Schurgers, O. Aberthorne, and M. Srivastava. Modulation Scaling for Energy Aware Communication Systems. In *International Symposium on Low Power Electronics and Design*, pages 96–99, 2001.