Everywhere Sparse Approximately Optimal Minimum Energy Data Gathering and Aggregation in Sensor Networks

Konstantinos Kalpakis
Computer Science & Electrical Engineering Department
University of Maryland Baltimore County

We consider two related data gathering problems for wireless sensor networks (WSNs). The MLDA problem is concerned with maximizing the system lifetime T so that we can perform T rounds of data gathering with in–network aggregation, given the initial available energy of the sensors. The M^{2}EDA problem is concerned with minimizing the maximum energy consumed by any one sensor when performing T rounds of data gathering with in–network aggregation, for a given T.

We provide an effective algorithm for finding an everywhere sparse integral solution to the M^{2}EDA problem which is within a factor of $\alpha = 1 + 4n/T$ of the optimum, where n is the number of nodes. A solution is everywhere sparse if the number of communication links for any subset X of nodes is $O(X)$, in our case at most $4|X|$. Since often $T = \omega(n)$, we obtain the first everywhere sparse, asymptotically optimal integral solutions to the M^{2}EDA problem. Everywhere sparse solutions are desirable since then almost all sensors have small number of incident communication links and small overhead for maintaining state.

We also show that the MLDA and M^{2}EDA problems are essentially equivalent, in the sense that we can obtain an optimal fractional solution to an instance of the MLDA problem by scaling an optimal fractional solution to a suitable instance of the M^{2}EDA problem. As a result, our algorithm is effective at finding everywhere sparse, asymptotically optimal, integral solutions to the MLDA problem, when the initial available energy of the sensors is sufficient for supporting optimal system lifetime which is $\omega(n)$.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless network communications, network topology.*

General Terms: Algorithms, Design.

Additional Key Words and Phrases: energy management, in-network aggregation, lifetime maximization, sparsity communication topology, wireless sensor networks.

1. INTRODUCTION

Wireless sensor networks (WSNs) are expected to consist of numerous inexpensive micro–sensors [Kahn et al. 1999; Min et al. 2001; Rabaey et al. 2000], readily deployable in var-

Author's address: K. Kalpakis, Computer Science & Electrical Engineering Department, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250. kalpakis@csee.umbc.edu

ious physical environments to collect useful information (e.g. seismic, acoustic, medical and surveillance data) in a robust and autonomous manner. There are several obstacles that need to be overcome before this vision becomes a reality [Heinzelman et al. 1999] – see Akyildiz et al [Akyildiz et al. 2002] for a comprehensive survey of issues arising in WSNs. These obstacles are due to the limited energy, computing capabilities, and communication resources available to the sensors. Often, replenishing the energy of the sensors by replacing their batteries is cost prohibitive or even infeasible.

Consider a WSN, where sensors have some initial non–replenishable energy. A basic operation in such a system is the systematic gathering of sensed data at a base station for processing. The key challenge is conserving the sensor energy so as to maximize the system's lifetime, the time until the first sensor depletes its energy. Energy–aware routing and data gathering have been the subject of intensive research, such as [Hou et al. 2006; Luo and Hubaux 2005; Pan et al. 2005; Sankar and Liu 2004; Yu et al. 2006], including the seminal work of Chang and Tassiulas [Chang and Tassiulas 2004]. Data gathering does not perform in-network data aggregation or fusion, which is a useful paradigm for increasing the system's lifetime, since it leads to significant energy savings [Krishnamachari et al. 2002; Madden et al. 2002; Madden et al. 2002]. In–network aggregation allows sensors to aggregate multiple input data packets into one output data packet; often, in each round, a sensor aggregates all the data packets it receives with its own data packet. In-network aggregation is most useful in computing for each round various statistical descriptors of the sensor measurements, such as the minimum, maximum, average, variance, approximate histogram, uniform fixed-size sample, measurements of high frequency, sketches, etc (e.g. see [Puttagunta and Kalpakis 2007]). Several important energy–aware protocols for data gathering with in–network data aggregation in WSNs have been proposed [Bhardwaj et al. 2001; Heinzelman et al. 2000; Kalpakis et al. 2003; Lindsey and Raghavendra 2002; Lindsey et al. 2001; Liu et al. 2004]. These protocols do not provide guarantees on their performance with respect to the optimal system lifetime.

We are concerned with the energy–efficient data gathering with in–network aggregation problem in WSNs. The maximum lifetime data gathering with in–network data aggregation (MLDA) problem is reduced in [Kalpakis et al. 2003] to the following directed network design problem: provision integral non–negative capacity c_e for each edge e of the network in order to maximize T (the lifetime), such that each sensor–base station directed vertex cut has capacity at least T, while the total energy consumed by each sensor in sending/receiving packets along its incident edges does not exceed its available energy. An edge $e = (u, v)$ with capacity c_e indicates that u sends at most c_e packets to v during the lifetime of the system, and thus imposes an energy demand on its head/tail nodes. Provisioning edge capacities so that each sensor can have maximum flow of at least T to the base station is both necessary and sufficient to achieve lifetime T [Kalpakis et al. 2003]. Even though this network design problem is NP–hard, heuristics in [Kalpakis et al. 2003] show that tight approximate solutions to the network design problem can be obtained in reasonable time for small to medium–size networks.

An alternative approach to tackling the energy–efficient data gathering with in–network aggregation in WSNs, is as follows. Minimize the maximum energy consumed by any sensor, while requiring that all sensor–base station directed vertex cuts have capacity at least a given constant T. For brevity, we call this problem the MINIMUM MAXIMUM ENERGY DATA GATHERING AND AGGREGATION (M^2EDA) problem.

A solution to an instance of the M^{2}EDA/MLDA problem is *everywhere sparse* if there are $O(X)$ positive capacity edges among the nodes in any subset X of nodes. A solution is integral if all edge capacities are integral, otherwise it is a *fractional* solution.

The original contributions of this paper are as follows.

—We show that there exist optimal fractional solutions to the M^{2}EDA problem such that there are at most $4|X|$ communication links among the nodes in any subset X of nodes, i.e. each M^{2}EDA instance has an optimal everywhere sparse fractional solution.
In other words, optimal lifetime data gathering with in–network aggregation can be done with everywhere sparse communication links. Consequently, almost all sensors have small number of incident communication links, and the overhead for each sensor for maintaining state information for data gathering with in–network aggregation is small.

—We provide an effective algorithm, ALGM^{2}EDA, for finding such optimal, everywhere sparse, fractional solutions to the M^{2}EDA problem.

—We show that by rounding down the solutions above, we obtain everywhere sparse integral solutions to the M^{2}EDA problem which are within a factor of α of the optimum, and where $\alpha = 1 + 4n/T$, T is the required lifetime, and n is the number of nodes. For $T = \omega(n)$, we have asymptotically optimal everywhere sparse integral solutions to the M^{2}EDA problem.

—We show that the M^{2}EDA and MLDA problems are essentially equivalent.

—We provide sparsity results and approximation bounds for the MLDA problem that are analogous to those given for the M^{2}EDA problem.

To the best of our knowledge, this is the first time that the sparsity structure of data–gathering with in–network aggregation in wireless sensor networks has been analyzed and utilized to obtain effective algorithms to find sparse approximately–optimal solutions.

We also present extensive experimental results demonstrating that the ALGM^{2}EDA algorithm routinely finds almost optimal everywhere sparse integral solutions to random instances of the M^{2}EDA and MLDA problems with ≤ 70 nodes in less than 30 minutes CPU time, improving substantially upon existing algorithms for these two problems.

The rest of the paper is organized as follows. Preliminaries are given in section 2. In sections 3 and 4 we analyze the sparsity structure of optimal solutions to the M^{2}EDA problem. We present our ALGM^{2}EDA algorithm for finding α–optimal everywhere sparse integral solutions to the M^{2}EDA problem in section 5. The relationship between the M^{2}EDA and MLDA problems is analyzed in section 6, and show how to obtain asymptotically optimal everywhere sparse solutions to the MLDA problem in section 7. We present experimental results in section 8, and conclude in section 9.

2. PRELIMINARIES

We provide definitions, notations, and other preliminaries we use in the rest of the paper.

2.1 Common definitions and notations

Given a directed graph G, we denote its vertex and edge sets with $V[G]$ and $E[G]$ respectively. For brevity, we often write $v \in G$ instead of $v \in V[G]$ for a vertex v, and $ij \in G$ instead of $ij \in E[G]$ for an edge ij. A subset of vertices $S \subseteq V[G]$ induces the subgraph $G[S] = (S, E[S])$ of G where $E[S] = E[G] \cap S \times S$. Let $\delta^{\mathrm{o}}(S) \subseteq E[G]$ be the set of edges leaving S, $\delta^{\mathrm{i}}(S) \subseteq E[G]$ be the set of edges entering S, and $\delta(S_1, S_2)$ be the set

of edges that leave S_1 and enter S_2, where $S, S_1, S_2 \subseteq V[G]$. Note that $\delta^{\mathrm{i}}(S) = \delta^{\mathrm{o}}(\overline{S})$ for all $S \subseteq V[G]$, where $\overline{S} = V[G] - S$ is the complement of S. A *directed cut* is a partition $(\overline{S}, S)$ of $V[G]$. For brevity, we denote a directed cut $(\overline{S}, S)$ simply with the set S of vertices it enters. [1] An edge $e \in E[G]$ *crosses a directed cut* S if $e \in \delta^{\mathrm{i}}(S)$. A graph G is *sparse* iff $|E[G]| = O(V[G])$, and it is *everywhere sparse* iff for each $S \subseteq V[G]$ the induced subgraph $G[S]$ is sparse.

Given an instance $\mathfrak{I}$ of a discrete optimization problem, let $\textsc{opt}(\mathfrak{I})$ and $\textsc{sol}(\mathfrak{I})$ be an optimal and a feasible solution to $\mathfrak{I}$, respectively. For brevity, $\textsc{opt}(\mathfrak{I})$ and $\textsc{sol}(\mathfrak{I})$ will also indicate the value of the corresponding solution. The relative error of a solution $\textsc{sol}(\mathfrak{I})$ is equal to $|\textsc{opt}(\mathfrak{I}) - \textsc{sol}(\mathfrak{I})| / \textsc{opt}(\mathfrak{I})$ and its approximation ratio is equal to $\textsc{sol}(\mathfrak{I})/\textsc{opt}(\mathfrak{I})$. An instance $\mathfrak{I}$ of a discrete optimization problem can be relaxed by allowing fractional values for its unknowns; a solution $\mathcal{S}$ to such a relaxed $\mathfrak{I}$ is called a *fractional* solution of $\mathfrak{I}$. If $\mathcal{S}$ assigns to each unknown an integer value then it is called an *integral* solution of $\mathfrak{I}$.

We denote vectors and matrices with lower and upper case bold letters, e.g. $\mathbf{x}_{n\times 1}$ and $\mathbf{A}_{n\times m}$ is an n–dimensional vector and an $n \times m$ matrix respectively. For brevity we omit the vector and matrix dimensions when they are clear from the context. Given an $n \times m$ matrix $\mathbf{A}_{n\times m}$ and two index sequences $I \subseteq \{1, 2, \ldots, n\}$ and $J \subseteq \{1, 2, \ldots, m\}$, we denote by $\mathbf{A}_{I,J}$ the sub–matrix $\mathbf{X}_{|I|\times|J|} = (x_{ij})$ of $\mathbf{A}$ where $x_{ij} = a_{I_i,J_j}$, and by $\mathbf{A}_J$ the sub–matrix $\mathbf{X}_{n\times|J|} = (x_{ij})$ of $\mathbf{A}$, where $x_{ij} = a_{iJ_j}$. For brevity, we indicate the ith column of $\mathbf{A}$ with $\mathbf{A}_i$. We denote the transpose of a matrix $\mathbf{A}$ with $\mathbf{A}^T$. The support of a vector $\mathbf{x}$ is defined as $\mathbb{I}(\mathbf{x}) = \{i \ : \ x_i \neq 0\}$. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$, we say that $\mathbf{x}$ *dominates* $\mathbf{y}$ and write $\mathbf{x} \geq \mathbf{y}$, if $x_i \geq y_i$ for $i = 1, 2, \ldots, m$. We say that $\mathbf{x}$ is *lexicographically larger (smaller)* than $\mathbf{y}$ if the smallest index non–zero component of $\mathbf{x} - \mathbf{y}$ is positive (negative).

Consider a universe (set) U. A function $f : U \longrightarrow \mathbb{R}$ can be viewed as a vector $\mathbf{f} \in \mathbb{R}^{|U|}$ and vice versa. The support of a function f is defined as the support of its corresponding vector $\mathbf{f}$. Let $f(X) = \sum_{x\in X} f(x)$ be the extension of function f to subsets $X \subseteq U$. Given two functions $f, g : U \longrightarrow \mathbb{R}$, we say that f *dominates* g iff $\mathbf{f}$ dominates $\mathbf{g}$. The characteristic vector $\chi(U') \in \{0, 1\}^{|U|}$ of any subset $U' \subseteq U$ is such that $\chi_u = 1$ iff $u \in U'$.

We provide an overview of some concepts in linear programming we will use later on in Appendix A. The reader is referred to a text such as [Korte and Vygen 1991; Bertsimas and Tsitsiklis 1997] for further details.

2.2 A model for wireless sensor networks

Consider a wireless sensor network (WSN) with n nodes. One node, denoted b, is designated as the base station, with the remaining nodes being sensors. Sensors are identified by unique IDs and are assumed to have limited non–replenishable (initial) energy while the base station has no energy limitations. Time is discrete, and at each time period, we are interested in gathering the values from all the sensors to the base station, and more precisely computing some function of all these values at the base station. During data gathering, in–network aggregation is assumed, i.e. any number of incoming data packets at a node

[1] Note that S denotes both a directed cut and a set of vertices. We write $b \in S$ to mean that b belongs in the set of vertices S.

can be aggregated into a single outgoing data packet. [2] Data packets have fixed size. The system lifetime T is the earliest time at which one or more sensors deplete their energy.

The topology of the wireless sensor network is modeled by a directed graph (digraph) G, with $V[G] = \{1, 2, \ldots, n\}$ and $E[G] \subseteq V[G] \times V[G]$. Often G is dense, i.e. $|E[G]| = \Theta(V[G]^2)$. There exists an edge $ij \in E[G]$ whenever i can successfully send a packet to j. Let τ_{ij} be the energy consumed by node i in order to transmit a single packet to node j, and let r_j be the energy needed to receive such a packet at node j. Often τ_{ij} is a monotonically non–decreasing function of the Euclidean distance between nodes i and j, i.e. the same or more energy is required to transmit a packet at longer distances. Let ϵ_i be the energy available at node i. We assume, w.l.o.g, that $\epsilon_b = \infty$, $r_b = 0$, and $\tau_{bi} = 0, \forall i \in V[G]$. For simplicity we consider sensor networks with a single base station. Sensor networks with multiple base stations can be handled easily — introduce a new node, b', to serve as the new single base station, and then append to G, for each current base station i, a new edge ib' with $\tau_{ib'} = 0$.

Let $c : E[G] \longrightarrow \mathbb{R}$ be a non–negative edge capacity function for the network G. For each edge $ij \in E[G]$, node i is permitted to send a total of c_{ij} packets to node j throughout the lifetime of the system, consuming energy equal to

$$\mu_i(\mathbf{c}) = \sum_{ij \in E[G]} \tau_{ij} c_{ij} + \sum_{ji \in E[G]} r_i c_{ji}. \tag{1}$$

Let $\mu_{\max}(\mathbf{c})$ be the maximum energy consumed by any sensor under the capacity function $\mathbf{c}$. We call a sensor i *tight* if $\mu_i(\mathbf{c}) = \mu_{\max}(\mathbf{c})$. An edge capacity function $\mathbf{c}$ induces the subgraph $G^*_{\mathbf{c}} = (V[G], E^*_{\mathbf{c}})$ of G, where $E^*_{\mathbf{c}}$ consists of all edges $e \in E[G]$ that have positive capacity c_e.

We extend the capacity function $\mathbf{c}$ to directed cuts $S \subseteq V[G]$ of G by setting the capacity $c(S)$ of S equal to the sum of the capacities of the edges in $E[G]$ entering S. For brevity, we denote this induced cut capacity function also by $\mathbf{c}$.

A demand function $\boldsymbol{\rho} : 2^{V[G]} \longrightarrow \mathbb{R}$ specifies a lower bound on the required capacity of the directed cuts of G. A directed cut S is *tight* if its capacity $c(S)$ is equal to its demand $\rho(S)$. We say that $\mathbf{c}$ *respects* a directed cut S iff $c(S) \geq \rho(S)$. We say that a $\mathbf{c}$ is *feasible* if it respects all the directed cuts of G, i.e. $\mathbf{c}$ dominates $\boldsymbol{\rho}$. The minimum positive demand of all the directed cuts is denoted with $\boldsymbol{\rho}_{\min} = \min\{\rho(S) \ : \ \rho(S) > 0 \text{ and } S \subseteq V[G]\}$.

We define the special demand function $\boldsymbol{\rho}_{b,T} : 2^{V[G]} \longrightarrow \mathbb{R}$ with parameters b and T as follows

$$\rho_{b,T}(S) = \begin{cases} T, & \text{if } S \subset V[G] \text{ and } b \in S, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This demand function is often used in practice for data gathering problems in WSNs, such as the MLDA problem (see section 6).

Data gathering with in–network aggregation problems in WSNs can be stated succinctly and analyzed by considering capacity functions that dominate certain demand functions [Kalpakis et al. 2003]; see also section 2.3 below.

[2] Incoming and outgoing packets carry concise fixed–size summaries of the sensor data [Puttagunta and Kalpakis 2007].

2.3 Data gathering with in–network aggregation problems

Consider a network G with base station b, transmit and receive energy consumption $\boldsymbol{\tau}$ and $\mathbf{r}$, and initial node energy $\boldsymbol{\epsilon}$. When performing data gathering with in–network aggregation in such a network, one is typically interested in maximizing the system lifetime T. This is known as the MAXIMUM LIFETIME DATA AGGREGATION (MLDA) problem. Kalpakis et al [Kalpakis et al. 2003] show that an instance $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem can be essentially solved by solving the following network design problem:

> given $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$, find an edge capacity function $\mathbf{c}$ that maximizes T while ensuring that $\mathbf{c}$ dominates $\boldsymbol{\rho}_{b,T}$ and $\boldsymbol{\epsilon}$ dominates $\mu(\mathbf{c})$.

Intuitively, if one can find an edge capacity function $\mathbf{c}$ such that (1) each sensor has a maximum–flow to the base station b of value at least T, and (2) each sensor has enough energy to send and receive all the packets provisioned by $\mathbf{c}$, then a lifetime T is possible; to show it is actually achievable additional work is needed. Kalpakis et al [Kalpakis et al. 2003] first find, in polynomial time, a fractional solution $\mathbf{c}$ to the MLDA network design problem by solving a linear program with $O(V[G]^3)$ constraints and unknowns; this linear program encodes the intuition given above. Second, Kalpakis et al [Kalpakis et al. 2003] show how to construct in polynomial–time a collection of *aggregation trees* (directed spanning trees rooted at the base station with all edges directed towards the root) and determine the number of rounds each one will be used to achieve optimal lifetime T for the data gathering task with in–network aggregation. An example network and its aggregation trees for the MLDA problem is given in Fig. 1.

Our focus in this paper is the problem of minimizing the maximum energy used by any sensor during data gathering with in-network aggregation for a given lifetime T. We call this the MINIMUM MAXIMUM ENERGY DATA AGGREGATION (M^2EDA) problem. As in [Kalpakis et al. 2003], the M^2EDA problem reduces to the following M^2EDA network design problem with $\boldsymbol{\rho} = \boldsymbol{\rho}_{b,T}$:

> given $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \,\rangle$, find an edge capacity function $\mathbf{c}$ that minimizes $\mu_{\max}(\mathbf{c})$ while ensuring that $\mathbf{c}$ dominates $\boldsymbol{\rho}$.

Though often in practice we have the same demand T for all directed cuts that contain the base station, it is convenient to formally allow those directed cuts to have differing demands. Hereafter, unless specified otherwise, we assume that $\boldsymbol{\rho}$ is an arbitrary demand function.

We show later that when the demand function $\boldsymbol{\rho}$ in M^2EDA is equal to the special demand function $\boldsymbol{\rho}_{b,T}$, the M^2EDA and MLDA problems are essentially equivalent.

We say that a feasible solution $\langle\, \mu_{\max}(\mathbf{c}), \mathbf{c} \,\rangle$ to an instance $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \,\rangle$ of the M^2EDA problem is *sparse, everywhere sparse, or dense* if the induced graph $G^*_{\mathbf{c}}$ is sparse, everywhere sparse, or dense, respectively. Similarly, we say that a feasible solution $\langle\, T, \mathbf{c} \,\rangle$ to an instance $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem is *sparse, everywhere sparse, or dense* if the induced graph $G^*_{\mathbf{c}}$ is sparse, everywhere sparse, or dense, respectively.

3. PRIMAL LINEAR PROGRAM FOR THE M^2EDA PROBLEM

We provide a mixed–integer programming formulation and its linear programming relaxation for finding optimal integral (fractional) solutions to the M^2EDA problem, respectively.

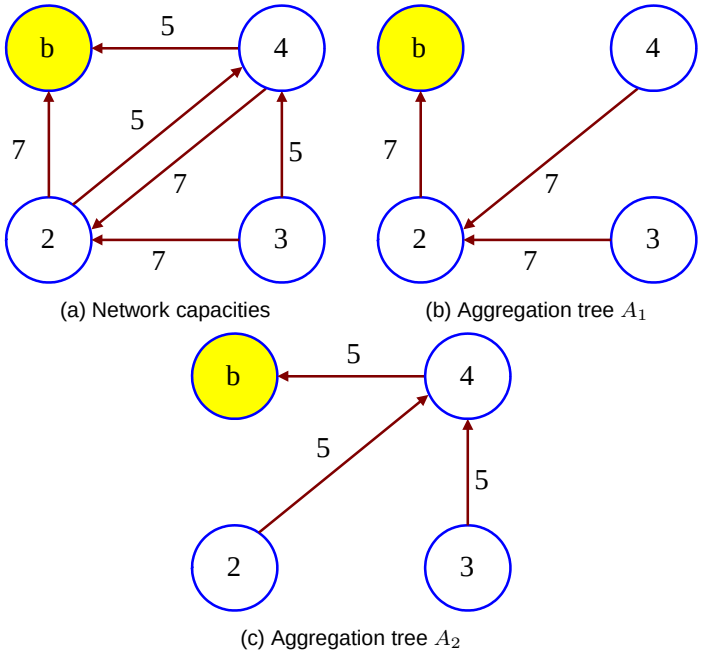


Fig. 1. Provisioned capacities for a WSN with lifetime 12 rounds, and two aggregation trees A_1 and A_2 with lifetimes 7 and 5 rounds respectively. For example, in tree A_1, for each one of 7 rounds, node 2 aggregates its own value with those it receives from nodes 3 and 4, and sends the result to the base station b.

Consider an instance $\mathfrak{I} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the M^{2}EDA network design problem. An optimal integral (fractional) solution $\text{OPT}(\mathfrak{I})$ to $\mathfrak{I}$ can be obtained by solving the following mixed integer (linear) program

$$\text{MIP1 (LP1):} \left\{ \begin{array}{ll} \min \mu & \text{such that} \\ \mu \geq \mu_i(\mathbf{c}), & \forall i \in V[G] \\ c(S) \geq \rho(S), & \forall S \subseteq V[G] \end{array} \right\} \tag{3}$$

where all the edge capacities c_{ij} are non–negative integers (reals). The constraints on μ account for the energy consumed by the nodes, since $\mu_i(\mathbf{c})$ is the energy consumed by a node i due to sending and receiving the number of packets provisioned by $\mathbf{c}$. The constraints $c(S) \geq \rho(S)$ for each directed cut S ensure that $\mathbf{c}$ dominates $\boldsymbol{\rho}$, as required for any feasible $\mathbf{c}$ in the M^{2}EDA network design problem.

The linear programming relaxation LP1 of MIP1 provides an optimal fractional solution and a lower bound on the optimal integral solution to the M^{2}EDA problem. We call LP1 the primal linear program for M^{2}EDA. Observe that feasible solutions to LP1 scale linearly as we scale the demand function $\boldsymbol{\rho}$. For simplicity, we often use $\mathbf{c}$ when referring to a basic feasible solution (or extreme point) of LP1, since there is 1–to–1 correspondence between the edge capacity function $\mathbf{c}$ and the extreme points $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ of the polytope defined by LP1.

It is impractical to solve the linear program LP1 by explicitly enumerating all its constraints even for small networks G, since there are $\Theta(2^{V[G]})$ constraints. Fortunately, we do not need to explicitly enumerate all these exponentially many constraints in solving it, since, as it is shown next, there exists polynomial–time algorithms to find any violated constraints and to solve LP1. [3] Unfortunately, this algorithm of solving M^{2}EDA, which we call ELLIPSOIDM^{2}EDA, is often impractical.

THEOREM 3.1. *LP1 has a polynomial–time separation oracle and thus it can be solved within any required accuracy in polynomial time.*

PROOF. There exists a polynomial–time separation oracle for the M^{2}EDA problem that allows us to check whether any edge capacity assignment $\mathbf{c}$ is feasible. A capacity assignment $\mathbf{c}$ is feasible, unless there exists a node $i \in V[G]$, with a minimum i–b directed cut $S \subseteq V[G]$ whose capacity $c(S)$ is less than its demand $\rho(S)$. For example, to find such a node i, we could simply solve an i–b max–flow problem for G with the given edge capacities $\mathbf{c}$. Since the oracle performs $O(V[G])$ max–flow computations on G, its worst–case running time is $O(\min\{V[G]^2 E[G], V[G]^3\})$.

Using the separation oracle above and the ellipsoid–based algorithm of Grötschel, Lovász, and Schrijver [Grötschel et al. 1981], we can find in polynomial–time a feasible solution to LP1 whose value is within ϵ (the required accuracy) from the optimal value, and where the feasible solution found may be in the interior of the polytope defined by LP1. ■

The ellipsoid–based algorithm [Grötschel et al. 1981] assumes that the polytope of the linear program is bounded. Without loss of generality, we extend LP1 with additional constraints to ensure a bounded polytope $P \subset \mathbb{R}^d$, with $d = |E[G] + 1$, as follows: we require that $c_e \leq 2T$ for all edges $e \in E[G]$ and that $\mu \leq 2T\mu_{\max}(\mathbf{1})$. The polytope P is contained in the ball with center at $(T+1)\mathbf{1}$ and radius $O(\sqrt{d}T\mu_{\max}(\mathbf{1}))$, and P contains the ball with center $(T+1)\mathbf{1}$ and radius $\Omega(\mu_{\max}(\mathbf{1})/\sqrt{d})$. From Theorem 4.19 in [Korte and Vygen 1991], it follows that the number of iterations of this ellipsoid–based algorithm as well as the number of calls to the oracle is $O(E[G]^2\beta)$, while the total running time is $O(E[G]^6\beta^2 + E[G]^4\beta \cdot \phi)$ where $\phi = O(\min\{V[G]^2 E[G], V[G]^3\})$ is the time for each oracle call, and $\beta = O(\log(T\mu_{\max}(\mathbf{1})E[G]/\epsilon))$. Since often $|E[G]| = \Theta(V[G]^2)$, it follows that the ellipsoid–based algorithm performs $O(V[G]^4 \log(V[G]))$ iterations/calls to the separation oracle, and its total running time is $O(V[G]^{12} \log^2(V[G]))$. The ellipsoid–based algorithm is impractical despite its polynomial worst–case running time.

4. M^{2}EDA HAS EVERYWHERE SPARSE OPTIMAL SOLUTIONS

We show that all instances $\langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the M^{2}EDA problem admit optimal fractional solutions that are everywhere sparse, i.e. solutions with $O(X)$ positive capacity edges among the nodes in any subset X of nodes. To this end, we utilize the primal formulation LP1 for the M^{2}EDA problem and arguments that are similar to those in Goemans [Goemans 2006]. Goemans [Goemans 2006] deals with cross–free families of intersecting sets, while we are dealing with cross–free families of crossing supermodular sets (see below) and additional constraints.

Since the M^{2}EDA problem admits everywhere sparse solutions, it follows that optimal lifetime data gathering with in–network aggregation can be done with everywhere sparse

[3] An algorithm that when it is given a candidate solution to a linear program, either finds some violated constraints or determines that none exist is called a separation oracle for that linear program.

communications. Consequently, almost all sensors have small degree, and the overhead for each sensor for maintaining state information for data gathering with in–network aggregation is small. Further, with $O(X)$ communication links among any subset X of nodes, we expect less contention for the limited bandwidth almost everywhere. Moreover, the maximum loss of lifetime due to rounding down a fractional solution to M^{2}EDA is bounded by $O(V[G])$ rather than the often much larger $O(E[G])$, leading to integral solutions with better approximation ratios.

4.1 Modularity properties of the demand and capacity of directed cuts

First, we need a few definitions. Consider any universe (set) U and a family (collection) $\mathcal{S}$ of subsets of U. We say that two subsets X and Y of U are *intersecting* iff all three subsets $X - Y$, $Y - X$, and $X \cap Y$ are non–empty. We say that two subsets X and Y of U are *crossing* if they are intersecting and $X \cup Y \neq U$. The family $\mathcal{S}$ is *cross–free* if it has no pair of crossing sets, and it is *laminar* if it has no pair of intersecting sets. [4] A function $f : 2^U \longrightarrow \mathbb{R}$ is called *(crossing, intersecting) supermodular* iff

$$f(X) + f(Y) \leq f(X \cup Y) + f(X \cap Y) \tag{4}$$

for all (crossing, intersecting) sets $X, Y \subseteq U$. The function f is called *submodular* if

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y), \tag{5}$$

for all sets $X, Y \subseteq U$. The interested reader is referred to Korte [Korte and Vygen 1991, Ch. 2] for more details.

Consider now a digraph G together with an edge capacity function $\mathbf{c}$ (extended to directed cuts) and a demand function $\boldsymbol{\rho}$ for G. Suppose that the universe U is equal to $V[G]$ and that the collection of sets $\mathcal{S}$ consists of all the directed cuts (subsets of vertices) of G. First, we show that $\mathbf{c}$ is submodular. Second, whenever $\boldsymbol{\rho}$ is crossing supermodular and $\mathbf{c} \geq \boldsymbol{\rho}$, we show that if two crossing directed cuts X and Y are tight then the directed cuts $X \cap Y$ and $X \cup Y$ are also tight. We will use the latter fact to construct a collection of non–crossing tight directed cuts that are essential in determining each extreme point of the polytope of LP1.

LEMMA 4.1. *The cut capacity function induced by any edge capacity function* $\mathbf{c}$ *is submodular.*

$$\begin{aligned} &c(X) + c(Y) - c(X \cup Y) - c(X \cap Y) = \\ &c(\delta(X - Y, Y - X)) + c(\delta(Y - X, X - Y)) \end{aligned} \tag{6}$$

for all subsets $X, Y \subseteq V$.

See Proposition 1.3 in Frank [Frank 1993] for a proof of Lemma 4.1.

LEMMA 4.2. *Consider any non-negative edge capacity function* $\mathbf{c}$ *and a crossing supermodular demand function* $\boldsymbol{\rho}$ *for a digraph* G. *Suppose* $\mathbf{c}$ *dominates* $\boldsymbol{\rho}$. *If any two crossing sets* $X, Y \subset V[G]$ *are tight then both* $X \cup Y$ *and* $X \cap Y$ *are tight, and in addition,*

$$c(X) + c(Y) = c(X \cup Y) + c(X \cap Y) \tag{7}$$

and $c(\delta(X - Y, Y - X)) = c(\delta(Y - X, X - Y)) = 0$.

[4] Two sets X and Y are non–crossing if one of the following is true: $X \subseteq Y$, $Y \subseteq X$, or X and Y are disjoint.

PROOF. Consider any two tight crossing sets $X, Y \subset V[G]$. The submodularity of $\mathbf{c}$ (due to Lemma 4.1) together with the tightness of X and Y imply that

$$\rho(X) + \rho(Y) = c(X) + c(Y) \geq c(X \cup Y) + c(X \cap Y). \tag{8}$$

By definition of crossing–supermodularity, and since $\boldsymbol{\rho}$ is crossing–supermodular, we have that

$$\rho(X \cup Y) + \rho(X \cap Y) \geq \rho(X) + \rho(Y). \tag{9}$$

Furthermore, from the two equations above, and since $\mathbf{c}$ dominates $\boldsymbol{\rho}$, it follows that both $X \cup Y$ and $X \cap Y$ are tight and

$$\rho(X \cup Y) + \rho(X \cap Y) = \rho(X) + \rho(Y). \tag{10}$$

Since the left hand–side of (6) is 0 and $\mathbf{c}$ is non–negative, $c(\delta(X - Y, Y - X)) = c(\delta(Y - X, X - Y)) = 0$. ■

4.2 LP1 has everywhere sparse extreme points

Consider an instance $\mathfrak{I} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the M^{2}EDA problem, where $\boldsymbol{\rho}$ is crossing supermodular. Let $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ be an extreme point (basic feasible solution or bfs) of the linear program LP1 for $\mathfrak{I}$. Recall that this extreme point is everywhere sparse iff the induced graph $G^*_{\mathbf{c}}$ is everywhere sparse. By the theory of linear programming [Bertsimas and Tsitsiklis 1997], we know that every bfs is uniquely determined by those constraints that become tight, i.e. are satisfied with equality, and their number is an upper bound on the size of the support (i.e. number of non-zero components) of each such bfs. Note that the support $\mathbb{I}(\mathbf{c})$ of $\mathbf{c}$ determines $E^*_{\mathbf{c}}$. We will show that $G^*_{\mathbf{c}}$ is everywhere sparse by carefully examining the structure of such tight constraints.

First, we need a few definitions. We define the characteristic energy vector $\hat{\mu}_i$ of node i to be a vector in $\mathbb{R}^{|E[G]|}$, whose e^{th} entry is equal to the energy consumed by node i due to the transmission or receipt of one packet along the edge $e \in E[G]$. Observe that the total energy $\mu_i(\mathbf{c})$ consumed by node i under a capacity function $\mathbf{c}$ is equal to $\mathbf{c}^T \cdot \hat{\mu}_i$.
Given a family of directed cuts $\mathcal{C} \subseteq 2^{V[G]}$ and a set of nodes $\mathbb{S} \subseteq V[G]$, let $\mathrm{rows}(\mathcal{C}, \mathbb{S})$ be the following collection of vectors: the characteristic vectors $\chi(\delta^{\mathrm{i}}(X))$ of the sets of edges $\delta^{\mathrm{i}}(X)$ entering each cut $X \in \mathcal{C}$, together with the characteristic energy vectors $\hat{\mu}_i$ of each node $i \in \mathbb{S}$. Let $\mathrm{span}(\mathcal{C}, \mathbb{S})$ be the vector space spanned by the vectors in $\mathrm{rows}(\mathcal{C}, \mathbb{S})$, and let $\dim(\mathcal{V})$ be the dimension of a vector space $\mathcal{V}$. Observe that $\mathrm{rows}(\mathcal{C}, \mathbb{S})$ essentially provide us all the constraints of the linear program LP1 due to the directed cuts in $\mathcal{C}$ and the nodes in $\mathbb{S}$.

We are now ready to examine the structure of the tight constraints of LP1. Let $\mathcal{C} = \{S : c(S) = \rho(S) > 0\}$ be the set of all tight directed cuts and $\mathbb{S} = \{i : \mu_i(\mathbf{c}) = \mu_{\max}(\mathbf{c})\}$ be the set of all tight nodes, with respect to the given bfs $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$. Note that $\mathcal{C}$ can not contain the empty set or any singleton set except possible for $\{b\}$. Our main goal is to find a "nice" basis of the vector space $\mathrm{span}(\mathcal{C}, \mathbb{S})$, which we accomplish in Lemma 4.3 below, and then bound the size of that basis.

LEMMA 4.3. *There exists a family $\mathcal{C}' \subseteq \mathcal{C}$ of cuts and a set $\mathbb{S}' \subseteq \mathbb{S}$ of nodes such that, $\mathcal{C}'$ is cross–free, the vector space $\mathrm{span}(\mathcal{C}', \mathbb{S}')$ equals the vector space $\mathrm{span}(\mathcal{C}, \mathbb{S})$ and has dimension $\dim(\mathrm{span}(\mathcal{C}', \mathbb{S}')) = |\mathcal{C}'| + |\mathbb{S}'|$, and the characteristic vectors of $\delta^{\mathrm{i}}(X)$, for each cut $X \in \mathcal{C}'$, are linearly independent.*

PROOF. First, we construct a cross–free family of cuts $\mathcal{C}'$ from $\mathcal{C}$, so that $\mathrm{span}(\mathcal{C}) = \mathrm{span}(\mathcal{C}')$ with all the characteristic vectors of $\delta^{\mathrm{i}}(S)$, $S \in \mathcal{C}'$, being linearly independent. Let $\mathcal{L}$ be a maximal collection of cross–free sets in $\mathcal{C}$. Such a collection $\mathcal{L}$ can be computed from $\mathcal{C}$ using the uncrossing algorithm in Hurkens et al [Hurkens et al. 1988].

We prove by contradiction that $\mathrm{span}(\mathcal{L}) = \mathrm{span}(\mathcal{C})$ and $|\mathcal{L}| = \dim(\mathrm{span}(\mathcal{C}))$. For the sake of contradiction, assume that $\mathrm{span}(\mathcal{L}) \neq \mathrm{span}(\mathcal{C})$. Then, there exists a set $S \in \mathcal{C}$ for which $\chi(\delta^{\mathrm{i}}(S)) \notin \mathrm{span}(\mathcal{L})$. Set S must cross at least one set already in $\mathcal{L}$, since $\mathcal{L}$ is maximal. Choose an $S \in \mathcal{C}$ that crosses the fewest elements in $\mathcal{L}$. Let $T \in \mathcal{L}$ be a set that S crosses. Since both S and T are in $\mathcal{C}$, they are both tight. By Lemma 4.2, the sets $S \cap T$ and $S \cup T$ are also tight, and thus both are in $\mathcal{C}$. We show that $S \cap T$ or $S \cup T$ cross fewer sets in $\mathcal{L}$, contradicting the choice of S above. Consider any set $X \in \mathcal{L}$. Clearly, X does not cross T since $\mathcal{L}$ is cross–free. There are four cases to consider.

—$X \subset T$. Since S crosses T, set X could only cross $S \cap T$. If it does, then $X \cap S, X - S, S - X, \overline{X \cap S} \neq \emptyset$, which implies that X crosses S as well.

—$T \subset X$. Since S crosses T, set X could only cross $S \cup T$. If it does, then $X \cap S, X - S, S - X, \overline{X \cap S} \neq \emptyset$, which implies that X crosses S as well.

—$X \cap T = \emptyset$. Since S crosses T, set X could only cross $S \cup T$. If it does, then X crosses S as well.

—$X \cup T = V$. In this case, X could only cross $S \cup T$. If it does, then $X - S, S - X, \overline{X \cup S} \neq \emptyset$, since S and T cross. Thus, X crosses S as well.

In all cases, $S \cap T$ or $S \cup T$ cross fewer sets in $\mathcal{L}$ than S crosses. Contradiction, since S is chosen to cross the fewest sets in $\mathcal{L}$.

Second, a construct $\mathbb{S}'$ from $\mathbb{S}$ such that $\mathrm{span}(\mathcal{C}, \mathbb{S}) = \mathrm{span}(\mathcal{C}', \mathbb{S}')$. Starting with $\mathbb{S}' = \emptyset$, consider each element of $\mathbb{S}$ in turn, and include it in $\mathbb{S}'$ until $\mathrm{span}(\mathcal{C}', \mathbb{S}') = \mathrm{span}(\mathcal{C}, \mathbb{S})$. ■

Lemma 4.3 implies that the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{y} :: \left\{ \begin{array}{l} \mathbf{x}(\delta^{\mathrm{i}}(S)) = \rho(S), \ \ \forall S \in \mathcal{C}' \\ \mu_i(\mathbf{x}) = \mu_{\max}(\mathbf{c}), \ \ \forall i \in \mathbb{S}' \end{array} \right\} \tag{11}$$

is of full–rank. Since $\mathrm{span}(\mathcal{C}, \mathbb{S}) = \mathrm{span}(\mathcal{C}', \mathbb{S}')$, the unique solution $\mathbf{x}$ of the linear program in Eq. (11) is equal to $\mathbf{c}$. Since the support size of $\mathbf{x}$ is $\leq \mathrm{rank}(\mathbf{A})$ and $\mathrm{rank}(\mathbf{A}) = |\mathcal{C}'| + |\mathbb{S}'|$, it follows that $|\mathcal{C}'| + |\mathbb{S}'|$ is an upper bound on the support size of $\mathbf{c}$. Note that the size of $\mathbb{S}'$ is at most $V[G] - 1$. Using Lemma 4.4 with the set family $\mathcal{C}' \subseteq 2^{V[G]}$, we show that the size of $\mathcal{C}'$ is at most $3|V[G]| - 1$, and hence the support size of $\mathbf{c}$ is at most $4|V[G]| - 2$.

LEMMA 4.4. *Consider a set family $\mathcal{S}$ from a universe U. The size $|\mathcal{S}|$ is $\leq 2|U|$ or $\leq 4|U| - 2$ whenever $\mathcal{S}$ is laminar or cross–free, respectively. If $\mathcal{S}$ contains at most one singleton, then its size is $\leq |U| + 1$ or $\leq 3|U| - 1$ whenever it is laminar or cross–free, respectively.*

PROOF. The first part of the lemma, that the size of a set family $\mathcal{S}$ is $\leq 2|U|$ or $\leq 4|U| - 2$ whenever $\mathcal{S}$ is laminar or cross–free, follows from Corollary 2.15 in [Korte and Vygen 1991].

Consider the second part of the lemma. Suppose, w.l.o.g., that $\mathcal{S}$ contains one singleton. Consider the set family $\mathcal{S}'$ obtained from $\mathcal{S}$ by including all the singleton subsets of U. Clearly, $|\mathcal{S}'| = |\mathcal{S}| + |U| - 1$. Since $\mathcal{S}'$ is laminar if $\mathcal{S}$ is laminar, and $|\mathcal{S}'| \leq 2|U|$,

it follows that $|\mathcal{S}| \leq |U| + 1$. Similarly, since $\mathcal{S}'$ is cross–free if $\mathcal{S}$ is cross–free, and $|\mathcal{S}'| \leq 4|U| - 2$, it follows that $|\mathcal{S}| \leq 3|U| - 1$. ■
Moreover, as in Goemans [Goemans 2006], we show that $G^*_{\mathbf{c}}$ is sparse.

LEMMA 4.5. *The graph $G^*_{\mathbf{c}}$ has at most most $4|V[G]| - 2$ edges.*

PROOF. Since $|E^*_{\mathbf{c}}| = |\mathcal{C}'| + |\mathbb{S}'|$, $|\mathbb{S}'| \leq |V[G]| - 1$, and $|\mathcal{C}'| \leq 3|V[G]| - 1$, it follows that $|E^*_{\mathbf{c}}| \leq 4|V[G]| - 2 \leq 4|V[G]|$. ■
Furthermore, we show that $G^*_{\mathbf{c}}$ is everywhere sparse as well.

LEMMA 4.6. *The subgraph $G^*_{\mathbf{c}}[V']$ has at most $\leq 3|V'|$ edges, where V' is any proper subset of $V[G]$.*

PROOF. The proof is similar to the proof of Theorem 5 of Goemans [Goemans 2006]. Consider the linear system $\mathbf{Ax} = \mathbf{y}$ in (11). Let $\mathbf{B}$ be the submatrix of $\mathbf{A}$ that consists of the columns $\mathbf{A}_e$ of $\mathbf{A}$ that correspond to the positive capacity edges $e \in E^*_{\mathbf{c}}[V']$, which is equal to the set of edges of $G^*_{\mathbf{c}}[V']$. The $\mathrm{rank}(\mathbf{B})$ of $\mathbf{B}$ is equal to $|E^*_{\mathbf{c}}[V']|$ since $\mathbf{A}$ is of full–rank. Thus, it is sufficient to compute an upper bound on the rank of $\mathbf{B}$. To this end, we count the number of distinct non–zero rows of $\mathbf{B}$.

First, consider all the rows of $\mathbf{B}$ that correspond to the energy constraint equations $\mu_i(\mathbf{x}) = \mu_{\max}(\mathbf{c})$ for the nodes $i \in \mathbb{S}'$. Consider a row that corresponds to a node $i \in \mathbb{S}'$ which is not in V'. Since i is not in V', we have that $E^*_{\mathbf{c}}[V'] \cap (\delta^{\mathrm{i}}(\{i\}) \cup \delta^{\mathrm{o}}(\{i\})) = \emptyset$, which implies that this row of $\mathbf{B}$ is identically zero. Thus, $\mathbf{B}$ can have at most $|V'|$ such non–zero rows.

Second, consider all the rows of $\mathbf{B}$ that correspond to the demand/capacity constraints $c(S) = \rho(S)$, for cuts $S \in \mathcal{C}'$. Note that if $S \cap V' = \emptyset$ then $\delta^{\mathrm{i}}(S) \cap E^*_{\mathbf{c}} * [V'] = \emptyset$ and hence such rows are identically zero. Observe that the family of sets $\mathcal{L} = \{S \cap V' : S \in \mathcal{C}', S \cap V' \neq \emptyset\}$ is laminar, since $\mathcal{C}'$ is cross–free. By Lemma 4.4, we have that $|\mathcal{L}| \leq 2|V'|$. Thus, $\mathbf{B}$ can have at most $2|V'|$ such non–zero rows.
Thus, $\mathbf{B}$ has at most $|V'| + 2|V'| = 3|V'|$ non–zero rows, and hence $3|V'| \geq \mathrm{rank}(\mathbf{B}) = |E^*_{\mathbf{c}}[V']|$. ■
Lemmas 4.5 and 4.6 imply that each basic feasible solution of LP1 is everywhere sparse.

THEOREM 4.7. *Consider an instance $\mathfrak{I} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the* M^{2}EDA *problem, where the demand function $\boldsymbol{\rho}$ is crossing supermodular. Let $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ be any basic feasible solution to the linear program LP1 for $\mathfrak{I}$. Then, $G^*_{\mathbf{c}}[V']$ has at most $4|V'|$ edges for each $V' \subseteq V[G]$, i.e. $G^*_{\mathbf{c}}$ is everywhere sparse.*

PROOF. Follows from Lemmas 4.5 and 4.6. ■

4.3 Integral solutions from optimal fractional basic feasible solutions

We show that by a rounding down an optimal fractional basic feasible solution to LP1, we obtain an everywhere sparse integral feasible solution to LP1 and MIP1 for M^{2}EDA whose value is close to the optimal value.

THEOREM 4.8. *Consider an instance $\mathfrak{J} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the* M^{2}EDA *problem with a crossing–supermodular demand $\boldsymbol{\rho}$. Let* OPT$(\mathfrak{J}) = \langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ *be an optimal fractional basic feasible solution to $\mathfrak{J}$. Then, $\langle \mu_{\max}(\lfloor \mathbf{c} \rfloor), \lfloor \mathbf{c} \rfloor \rangle$ is an integral solution to $\mathfrak{J}$ which is optimal within a factor of $\alpha = 1 + 4|V[G]|/\boldsymbol{\rho}_{\min}$, and $G^*_{\hat{\mathbf{c}}}[V']$ has at most $4|V'|$ edges for each $V' \subseteq V[G]$.*

PROOF. Let $\langle\, \mu_{\max}(\mathbf{c}), \mathbf{c}\,\rangle$ be an optimal (fractional) basic feasible solution to the linear program LP1 for the following scaled M^{2}EDA instance $\mathfrak{J}_\alpha = \langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \alpha \cdot \boldsymbol{\rho}\,\rangle$. Consider the integral capacity function $\hat{\mathbf{c}}$ that assigns capacity $\hat{c}_e = \lfloor c(e) \rfloor$ to each edge $e \in E[G]$. Since for any $V' \subseteq V[G]$, $G^*_{\hat{\mathbf{c}}}[V'] \subseteq G^*_{\mathbf{c}}[V']$ and $G^*_{\mathbf{c}}[V']$ has at most $4|V'|$ edges, it follows that $G^*_{\hat{\mathbf{c}}}[V']$ also has at most $4|V'|$ edges. Since $\hat{\mathbf{c}} \leq \mathbf{c}$, for each node $i \in V[G]$, the energy $\mu_i(\hat{\mathbf{c}})$ it consumes under $\hat{\mathbf{c}}$ is at most the energy $\mu_i(\mathbf{c})$ it consumes under $\mathbf{c}$. Hence,

$$\mu_{\max}(\hat{\mathbf{c}}) \leq \mu_{\max}(\mathbf{c}). \tag{12}$$

Consider now any directed cut $S \subseteq V[G]$ with positive demand. Since $\mathbf{c}$ is a feasible edge capacity assignment for the scaled instance $\mathfrak{J}_\alpha$, and since, due to Theorem 4.7, there are at most $4|V[G]$ edges $e \in E[G]$ with positive capacity c_e, we have that

$$\hat{c}(S) \geq c(S) - 4|V[G]| \geq \alpha \cdot \rho(S) - 4|V[G]|. \tag{13}$$

By the definition of α, we have

$$\alpha \cdot \rho(S) = \rho(S) + 4|V[G]| \cdot \frac{\rho(S)}{\boldsymbol{\rho}_{\min}} \geq \rho(S) + 4|V[G]|, \tag{14}$$

which together with (13) implies that $\hat{c}(S) \geq \rho(S)$. Therefore, $\langle\, \mu_{\max}(\hat{\mathbf{c}}), \hat{\mathbf{c}}\,\rangle$ is a feasible integral solution to the M^{2}EDA instance $\mathfrak{J}$. Since $\langle\, \mu_{\max}(\mathbf{c}), \mathbf{c}\,\rangle$ is an optimal fractional solution to $\mathfrak{J}_\alpha$, we have that $\langle\, \alpha^{-1}\mu_{\max}(\mathbf{c}), \alpha^{-1}\mathbf{c}\,\rangle$ is an optimal fractional solution to $\mathfrak{J}$,

$$\text{OPT}(\mathfrak{J}) = \alpha^{-1}\mu_{\max}(\mathbf{c}). \tag{15}$$

Therefore

$$\text{OPT}(\mathfrak{J}) \leq \mu_{\max}(\hat{\mathbf{c}}) \leq \mu_{\max}(\mathbf{c}) = \alpha \cdot \text{OPT}(\mathfrak{J}), \tag{16}$$

and $\langle\, \mu_{\max}(\hat{\mathbf{c}}), \hat{\mathbf{c}}\,\rangle$ is an an α–optimal, everywhere sparse, integral solution to $\mathfrak{J}$. ■

5. FINDING α–OPTIMAL EVERYWHERE SPARSE INTEGRAL SOLUTIONS FOR M^{2}EDA

Consider an instance $\mathfrak{J} = \langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho}\,\rangle$ of the M^{2}EDA problem, where $\boldsymbol{\rho}$ is crossing supermodular.

There is a need for a practical algorithm for finding everywhere sparse solutions to the M^{2}EDA problem. Theorems 4.7 and 4.8 require an optimal basic feasible solution (bfs) to LP1. Recall that the (impractical) ELLIPSOIDM^{2}EDA algorithm finds in polynomial–time an approximately–optimal fractional feasible solution p to LP1. Moreover, given a linear program with a separation oracle and an optimal solution to it, Jain shows how to obtain in polynomial–time an optimal basic feasible solution to that linear program [Jain 2001][Lemma 3.3]. Therefore, though impractical, one can find an optimal fractional basic solution to LP1 in polynomial–time.

We present ALGM^{2}EDA, a practical algorithm for finding an optimal fractional basic feasible solution to the LP1 for any instance of the M^{2}EDA problem. By rounding–down that fractional everywhere sparse solution as prescribed in Theorem 4.8, we obtain an everywhere sparse integral solution whose value is within a factor of $\alpha = 1 + 4|V[G]|/\boldsymbol{\rho}_{\min}$ of the optimal value. Since often in practice, the minimum positive demand $\boldsymbol{\rho}_{\min}$ is $\omega(V[G])$, ALGM^{2}EDA effectively provides asymptotically optimal, everywhere sparse integral solutions to the M^{2}EDA problem with crossing–supermodular demands.

ALGM^{2}EDA is an iterative algorithm whose pseudocode is given in Algorithm 1. The algorithm maintains a set $\mathcal{C}$ of directed cuts, the set of the cut constraints of LP1 which are explicitly enumerated.

Initially, $\mathcal{C}$ contains a collection of $3|V[G]|$ directed cuts; these cuts are chosen since we have found them effective in solving M^{2}EDA instances where the nodes are uniformly placed on a line. It is possible to initialize $\mathcal{C}$ to also include the set of all the directed cuts generated by the separation oracle during the run of the ELLIPSOIDM^{2}EDA algorithm for LP1. Although this modification could lead to fewer iterations, it is costlier to implement and likely impractical due to the large running–time of the ELLIPSOIDM^{2}EDA (see the proof of Theorem 3.1).

At each iteration, ALGM^{2}EDA finds an optimal bfs $\mathbf{c}$ to the linear program

$$\langle \min \mu_{\max}(\mathbf{x}) \text{ such that } \mathbf{x}(S) \geq \rho(S),\ \forall S \in \mathcal{C} \rangle. \tag{17}$$

Then, it tests whether $\mathbf{c}$ is feasible for LP1 by attempting to find a directed cut S for which $c(S) < \rho(S)$. If no such cut S is found, the algorithm terminates with $\mathbf{c}$ as an optimal bfs to LP1. Otherwise, it appends S to $\mathcal{C}$ and continues with the next iteration.

The number of iterations of the while–loop in lines 4–14 of ALGM^{2}EDA is bounded by $|\mathcal{C}|$ at termination, and hence ALGM^{2}EDA always terminates. The worst–case number of iterations is $O(2^{V[G]})$; the number of iterations is at most 11 in all our experiments with random M^{2}EDA instances with up to 80 nodes (see section 8). For each iteration, the time for line 7 is determined by the solver used to solve the $|\mathcal{C}| \times |E[G]|$ linear program, while lines 8–12 take $O(\min\{V[G]^3 E[G], V[G]^4\})$ time.

Despite the fact that ALGM^{2}EDA has exponential worst–case running time, it routinely finds almost optimal everywhere sparse integral solutions to random M^{2}EDA instances with up to 70 nodes in 30 minutes or less —- much faster than what has been previously reported in [Kalpakis et al. 2003; Stanford and Tongngam 2006] for the corresponding MLDA problems. Note that the algorithms in [Kalpakis et al. 2003; Stanford and Tongngam 2006] have polynomial worst–case running times.

The algorithm in [Kalpakis et al. 2003] uses a linear program with $O(V[G]^3)$ constraints and unknowns, where for typically random networks G most of the constraints are dense. Solving such large dense linear programs using state–of–the art interior–point methods is quite slow. The algorithm in [Stanford and Tongngam 2006], which is based on an iterative algorithm by Garg and Konemann [Garg and Konemann 1998], performs an excessive number of iterations (due to slow convergence for small approximation errors), where each iteration takes $O(V[G]^3)$ time to find a minimum cost branching of G with varying edge costs. [Kalpakis et al. 2003] does not give any theoretical guarantees of optimality or near–optimality of their solutions, while [Stanford and Tongngam 2006] does not provide any guarantees of sparsity.

6. ON THE EQUIVALENCE OF THE MLDA AND M^{2}EDA PROBLEMS

We show that an optimal fractional solution to an instance of the MLDA problem can be easily obtained from an optimal fractional solution to a suitable instance of the M^{2}EDA problem and vice versa.

First, we show that we can optimally solve any MLDA problem instance by solving a certain M^{2}EDA problem instance, which is constructed from the given MLDA instance.

LEMMA 6.1. *Let $\mathfrak{I} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \rangle$ be an instance of the* MLDA *problem, and let T*

Algorithm 1 The ALGM^{2}EDA algorithm for solving the M^{2}EDA problem.

Input: M^{2}EDA instance $\mathfrak{I}\langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$
Output: An optimal fractional basic feasible solution $\text{OPT}(\mathfrak{I}) = \langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$.

```
1. // Initialize C to a set with a few select directed cuts
2. rename the nodes so that node 1 becomes the base station b
3. C ⟵ all the directed cuts {i}, {i, i+1}, and {1, 2, ..., i}, ∀i ∈ V[G]
4. while true do
5.    // Construct or update, and then solve the linear program for M²EDA
6.    let Q(C) be the LP⟨ min μ_max(x) such that x(S) ≥ ρ(S), ∀S ∈ C ⟩
7.    ⟨ μ_max(c), c ⟩ ⟵ optimal bfs for Q(C)
8.    // Find violated constraints, if any
9.    for each node i ∈ V[G] do
10.      S ⟵ an i–b min–cut for G with c
11.      if c(S) < ρ(S) then
12.         append S to C
13.   if no change to C then
14.      return ⟨ μ_max(c), c ⟩
```

be any positive constant. Consider the M^{2}EDA *instance* $\mathfrak{J} = \langle G, b, \hat{\boldsymbol{\tau}}, \hat{\mathbf{r}}, \boldsymbol{\rho}_{b,T} \rangle$*, where* $\hat{\tau}_{ij} = \tau_{ij}/\epsilon_i$ *and* $\hat{r}_i = r_i/\epsilon_i$ *for all nodes* $i, j \in V[G]$*. If* $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ *is an optimal fractional solution to the* M^{2}EDA *instance* $\mathfrak{J}$*, then* $\langle \eta \cdot T, \eta \cdot \mathbf{c} \rangle$ *is an optimal fractional solution to the* MLDA *instance* $\mathfrak{I}$*, where* $\eta = 1/\mu_{\max}(\mathbf{c})$*.*

PROOF. Consider an optimal fractional solution $\text{OPT}(\mathfrak{J}) = \langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ to the M^{2}EDA instance $\mathfrak{J}$. Note that $\mu_{\max}(\mathbf{c})$ is positive. Let $\eta = 1/\mu_{\max}(\mathbf{c})$. Clearly, $\mu_{\max}(\eta\mathbf{c}) = 1$.

Observe that $\langle \mu_{\max}(\eta\mathbf{c}), \eta\mathbf{c} \rangle$ is an optimal fractional solution to the M^{2}EDA instance $\mathfrak{J}' = \langle G, b, \hat{\boldsymbol{\tau}}, \hat{\mathbf{r}}, \boldsymbol{\rho}_{b,\eta T} \rangle$. Since $\mu_{\max}(\eta\mathbf{c}) \leq 1$, it follows that $\langle \eta T, \eta\mathbf{c} \rangle$ is a feasible solution to the MLDA instance $\mathfrak{I}$, because no node consumes more than its initial energy.

We prove by contradiction that $\langle \eta T, \eta\mathbf{c} \rangle$ is an optimal fractional solution to the MLDA instance $\mathfrak{I}$. Consider an optimal fractional solution $\text{OPT}(\mathfrak{I}) = \langle T', \mathbf{c}' \rangle$ to the MLDA instance $\mathfrak{I}$, with $T' > \eta T$. Let $\beta = T/T'$. Since $c'(S) \geq T'$ for all $S \subseteq V[G]$ that contain the base station b, it follows that $\langle \mu_{\max}(\beta\mathbf{c}'), \beta\mathbf{c}' \rangle$ is a feasible solution to the M^{2}EDA instance $\mathfrak{J}$. Since at least one node consumes all its energy in the MLDA optimal solution $\text{OPT}(\mathfrak{I})$, it follows that $\max\{\mu_i(\mathbf{c}')/\epsilon_i\} = 1$, for at least one node $i \in V[G]$. Hence, $\mu_{\max}(\beta\mathbf{c}') = \beta$. By the optimality of $\langle \mu_{\max}(\mathbf{c}), \mathbf{c} \rangle$ for the M^{2}EDA instance $\mathfrak{J}$, we have that $\beta \geq \mu_{\max}(\mathbf{c}) = \frac{1}{\eta}$, which implies that $\eta\beta \geq 1$. On the other hand, since $T' > \eta T$ and $\beta = T/T'$, we have that $\eta\beta < 1$. Contradiction. ■

Second, we show how to optimally solve any M^{2}EDA problem instance with demand $\boldsymbol{\rho}_{b,T}$ by solving a certain MLDA problem instance, which is constructed from the given M^{2}EDA instance.

LEMMA 6.2. *Let* T_o *be any positive constant, and let* $\mathfrak{J} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho}_{b,T_o} \rangle$ *be an instance of the* M^{2}EDA *problem. Consider the* MLDA *instance* $\mathfrak{I} = \langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \rangle$ *with* $\epsilon_i = 1$ *for all nodes* $i \in V[G] - \{b\}$*. If* $\langle T, \mathbf{c} \rangle$ *is an optimal fractional solution to the* MLDA *instance* $\mathfrak{I}$*, then* $\langle \mu_{\max}(\eta\cdot\mathbf{c}), \eta\cdot\mathbf{c} \rangle$ *is an optimal fractional solution to the* M^{2}EDA *instance* $\mathfrak{J}$*, where* $\eta = T_o/T$*.*

PROOF. Consider an optimal fractional solution $\text{OPT}(\mathfrak{I}) = \langle T, \mathbf{c} \rangle$ to the MLDA in-

stance $\mathfrak{I}$. Since one or more nodes consume all their initial energy, $\mu_{\max}(\mathbf{c}) = 1$. Let $\eta = T_o/T$. Since $\eta c(S) \geq \eta\rho_{b,T}(S) = \rho_{b,T_o}(S)$ for all $S \subseteq V[G]$, it follows that $\langle\, \mu_{\max}(\eta\mathbf{c}), \eta\mathbf{c} \,\rangle$ is a feasible solution to the instance M^{2}EDA instance $\mathfrak{J}$. Note that $\mu_{\max}(\eta\mathbf{c}) = \eta\mu_{\max}(\mathbf{c}) = \eta$.

We prove by contradiction that $\langle\, \mu_{\max}(\eta\mathbf{c}), \eta\mathbf{c} \,\rangle$ is an optimal fractional solution to the M^{2}EDA instance $\mathfrak{J}$. Suppose that there exists an optimal fractional solution $\textsc{opt}(\mathfrak{J}) = \langle\, \mu_{\max}(\mathbf{c}'), \mathbf{c}' \,\rangle$ to the M^{2}EDA instance $\mathfrak{J}$ such that $\mu_{\max}(\mathbf{c}') < \eta$. Let $\beta = 1/\mu_{\max}(\mathbf{c}')$. Note that $\mu_{\max}(\beta\mathbf{c}') = 1$. Then, $\langle\, \beta T_o, \beta\mathbf{c}' \,\rangle$ is a feasible solution to the MLDA instance $\mathfrak{I}$. Hence, $\beta T_o \leq T$, which implies that $\eta\beta \leq 1$. However, since $\eta > \mu_{\max}(\mathbf{c}) = 1/\beta$, we also have $\eta\beta > 1$. Contradiction. ■

7. ALMOST OPTIMAL EVERYWHERE SPARSE INTEGRAL SOLUTIONS FOR MLDA

Consider an instance $\mathfrak{I} = \langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem, and the instance $\mathfrak{J} = \langle\, G, b, \hat{\boldsymbol{\tau}}, \hat{\mathbf{r}}, \boldsymbol{\rho}_{b,T} \,\rangle$ of the M^{2}EDA problem, where T is any positive constant, and $\hat{\tau}_{ij} = \tau_{ij}/\epsilon_i$ and $\hat{r}_i = r_i/\epsilon_i$ for all nodes $i, j \in V[G]$. We find an optimal everywhere sparse solution to the MLDA instance $\mathfrak{I}$ by using Lemmas 6.1 and 6.2, together with an optimal everywhere sparse basic feasible solution to the LP1 for the M^{2}EDA instance $\mathfrak{J}$.

First, we establish that M^{2}EDA problem instances with demand function $\boldsymbol{\rho}_{b,T}$ admit optimal, everywhere sparse, fractional basic feasible solutions as stipulated by Theorem 4.7. To this end, it is sufficient to show that the demand function $\boldsymbol{\rho}_{b,T}$ is crossing supermodular.

LEMMA 7.1. *Let T be any positive constant. The demand function $\boldsymbol{\rho}_{b,T}$ is crossing supermodular.*

PROOF. Recall that for any $S \subseteq V[G]$, the demand $\rho_{b,T}(S)$ of S is equal to 0 unless S is proper and it contains the base station b, in which case it is equal to T. Let X, Y be two crossing subsets of $V[G]$. Since X and Y are crossing, we have that $X \cup Y \neq V[G]$. There are three cases to consider.

—Suppose that $\rho_{b,T}(X) = \rho_{b,T}(Y) = T$. The base station b is in $X \cap Y$. Further, since $X \cup Y \neq V[G]$, we have that $\rho_{b,T}(X \cup Y) = \rho_{b,T}(X \cap Y) = T$, and hence (4) holds.

—Suppose that $\rho_{b,T}(X) = T$ and $\rho_{b,T}(Y) = 0$. Since $X \cup Y \neq V[G]$, we have that $\rho_{b,T}(X \cup Y) = T$, and (4) holds.

—Suppose that $\rho_{b,T}(X) = \rho_{b,T}(Y) = 0$. Since $\rho_{b,T}(S) \geq 0$ for all $S \subseteq V[G]$, (4) holds.

■

Lemma 6.1 shows that an optimal solution to an MLDA problem instance $\mathfrak{I}$ can be obtained by appropriately scaling an optimal solution to the corresponding M^{2}EDA problem instance with demand function $\boldsymbol{\rho}_{b,T_o}$, for some positive constant T_o. Since scaling a capacity function preserves its support, it follows that the ALGM^{2}EDA algorithm can be used to construct an optimal, everywhere sparse, fractional solution $\langle\, T, \mathbf{c} \,\rangle$ to an MLDA problem instance $\mathfrak{I}$. Furthermore, by rounding down $\mathbf{c}$, we show that we get an everywhere sparse integral solution to the MLDA instance $\mathfrak{I}$ which is optimal within a factor of $1 - 4|V[G]|/T$.

THEOREM 7.2. *Consider an* MLDA *problem instance $\mathfrak{I} = \langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$. We can find an optimal fractional feasible solution $\langle\, T, \mathbf{c} \,\rangle$ to the* MLDA *instance $\mathfrak{I}$ such that $G^*_{\mathbf{c}}[V']$ has at most $4|V'|$ edges, for any $V' \subseteq V[G]$. If $\alpha = 1 - 4|V[G]|/T \geq 0$, then the*

capacity function $\hat{\mathbf{c}} = \lfloor \mathbf{c} \rfloor$ *provides an* α*–optimal integral solution to the* MLDA *instance* $\mathfrak{I}$*, such that* $G^*_{\hat{\mathbf{c}}}[V']$ *also has at most* $4|V'|$ *edges, for any* $V' \subseteq V[G]$.

PROOF. Let T' be any positive constant. Consider Lemma 6.1, and the M^{2}EDA instance $\mathfrak{J}$ that corresponds to the given MLDA instance $\mathfrak{I}$ and T'. In particular, $\mathfrak{J} = \langle G, b, \hat{\boldsymbol{\tau}}, \hat{\mathbf{r}}, \boldsymbol{\rho}_{b,T'} \rangle$ is the M^{2}EDA instance of interest, where $\hat{\tau}_{ij} = \tau_{ij}/\epsilon_i$ and $\hat{r}_i = r_i/\epsilon_i$ for all nodes $i, j \in V[G]$.

Let $\text{OPT}(\mathfrak{J}) = \langle \mu_{\max}(\mathbf{c}'), \mathbf{c}' \rangle$ be an optimal fractional basic feasible solution to LP1 for the M^{2}EDA instance $\mathfrak{J}$. Such a solution can be computed by the ALGM^{2}EDA algorithm. By Lemma 6.1, $\langle \eta \cdot T', \eta \cdot \mathbf{c}' \rangle = \langle T, \mathbf{c} \rangle$ is an optimal fractional feasible solution to the MLDA instance $\mathfrak{I}$, where $\eta = 1/\mu_{\max}(\mathbf{c}')$.

From Theorem 4.7 and since $\mathbf{c}$ and $\mathbf{c}'$ have the same support, we have that $G^*_{\mathbf{c}}[V'] = G^*_{\mathbf{c}'}[V']$ and $G^*_{\mathbf{c}}[V']$ has at most $4|V'|$ edges, for each $V' \subseteq V[G]$.

Consider now the integral capacity function $\hat{\mathbf{c}}$ such that $\hat{c}_e = \lfloor c(e) \rfloor$, for each edge $e \in E[G]$. Since $\hat{\mathbf{c}} \leq \mathbf{c}$, we have

$$\mu_i(\hat{\mathbf{c}}) \leq \mu_i(\mathbf{c}) \leq \epsilon_i \tag{18}$$

for each node $i \in V[G]$. Since $\mathbf{c} \geq \hat{\mathbf{c}}$, it follows that each $G^*_{\hat{\mathbf{c}}}[V']$ has $\leq 4|V'|$ edges, $V' \subseteq V[G]$.

Since there are at most $4|V[G]|$ edges $e \in E[G]$ with positive capacity c_e, it follows that

$$\hat{c}(S) \geq c(S) - 4|V[G]| \geq T - 4|V[G]|, \tag{19}$$

for all $S \subseteq V[G]$ with positive demand $\rho_{b,T'}(S)$.

Since $\hat{\mathbf{c}}$ dominates $\boldsymbol{\rho}_{b,T-4|V[G]|}$, we have that $\langle \lceil T - 4|V[G]| \rceil, \hat{\mathbf{c}} \rangle$ is an integral, feasible, everywhere sparse solution to the MLDA instance $\mathfrak{I}$. Since the optimal for $\mathfrak{I}$ is T, it is also optimal within a factor of $\alpha = (T - 4|V[G]|)/T = 1 - 4|V[G]|/T$. ■

8. EXPERIMENTS

We provide experimental results illuminating certain performance aspects of the ALGM^{2}EDA algorithm. All experiments were done in Matlab R11.1 running on a Windows XP desktop with a Pentium III 931Mhz processor and 512MB of RAM. We consider sensor networks in which the sensors are uniformly distributed in a $50m \times 50m$ field, and the base station is fixed at location $(0, 0)$. We generate 10 random networks of size n, for each $n = 10$, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, and 80. We use the first order radio model [Heinzelman et al. 2000; Kalpakis et al. 2003; Lindsey et al. 2001] as the energy model. A sensor consumes $50nJ/bit$ to run the transmitter or receiver circuitry and $100pJ/bit/m^2$ for the transmitter amplifier. The packet size is 1000 bits. The demand function used by all the M^{2}EDA problem instances is $\boldsymbol{\rho}_{b,T}$ with T fixed to 1000 rounds. We consider six performance metrics: the maximum energy $\mu_{\max}(\mathbf{c})$ used by any sensor; the number of iterations of the outer while–loop of the ALGM^{2}EDA algorithm; the number of cuts in $\mathcal{C}$ at termination; $|E^*_{\mathbf{c}}|$, the number of edges with positive capacity; the relative error err_{rel} of the value of the integral solution $\lfloor \mathbf{c} \rfloor$ with respect to the value of the optimal fractional solution $\mathbf{c}$, $err_{rel} = |\mu_{\max}(\lfloor \mathbf{c} \rfloor) - \mu_{\max}(\mathbf{c})|/\mu_{\max}(\mathbf{c})$; and the CPU running time in seconds.

We show the average of these six performance metrics of the ALGM^{2}EDA algorithm in Table I. Scatter plots for these performance metrics for each instance are shown in Fig 2, together with the average and standard deviation of these metrics for each network size.

Table I. Experimental average of select performance metrics for the ALGM^{2}EDA algorithm.

n	iters	final $\|\mathcal{C}\|$	$\mu_{\max}(\mathbf{c})$	$\|E^*_{\mathbf{c}}\|$	err_{rel}	time (secs)
10	3.60	32.50	0.1210	25.40	0.92%	8.50
15	5.10	53.60	0.1113	34.10	1.42%	25.60
20	5.40	75.50	0.1068	48.00	1.95%	51.30
25	5.80	94.10	0.1061	60.10	2.45%	87.27
30	5.00	106.60	0.1035	70.50	2.97%	87.33
35	5.20	129.30	0.1024	83.40	3.46%	221.95
40	4.60	144.00	0.1016	94.20	3.96%	288.78
45	5.90	168.80	0.1018	108.10	4.47%	475.76
50	6.20	185.80	0.1012	120.20	4.97%	790.88
60	5.50	226.50	0.1006	143.40	5.95%	1097.50
70	5.40	254.20	0.0998	171.00	6.95%	1773.20
80	5.30	289.00	0.0993	188.60	7.96%	3937.58
all	5.25	146.66	0.1046	95.58	3.95%	737.14

On average, the ALGM^{2}EDA algorithm terminates after seven iterations while considering less than 300 cuts, which is substantially smaller than the 2^{80} worst–case bound on $\mathcal{C}$ for the largest network size of 80. The average number of cuts in $\mathcal{C}$ at termination seems to grow linearly with the network size instead of exponentially.

The experimental relative error is a third or less of that predicted by Theorem 4.8. For example, for $n = 80$, the experimental $err_{rel} = 7.96\%$ vs the 32% upper–bound provided by Theorem 4.8. Though err_{rel} increases as n increases, this is expected, since for fixed T, the value of an optimal fractional solution to M^{2}EDA decreases.

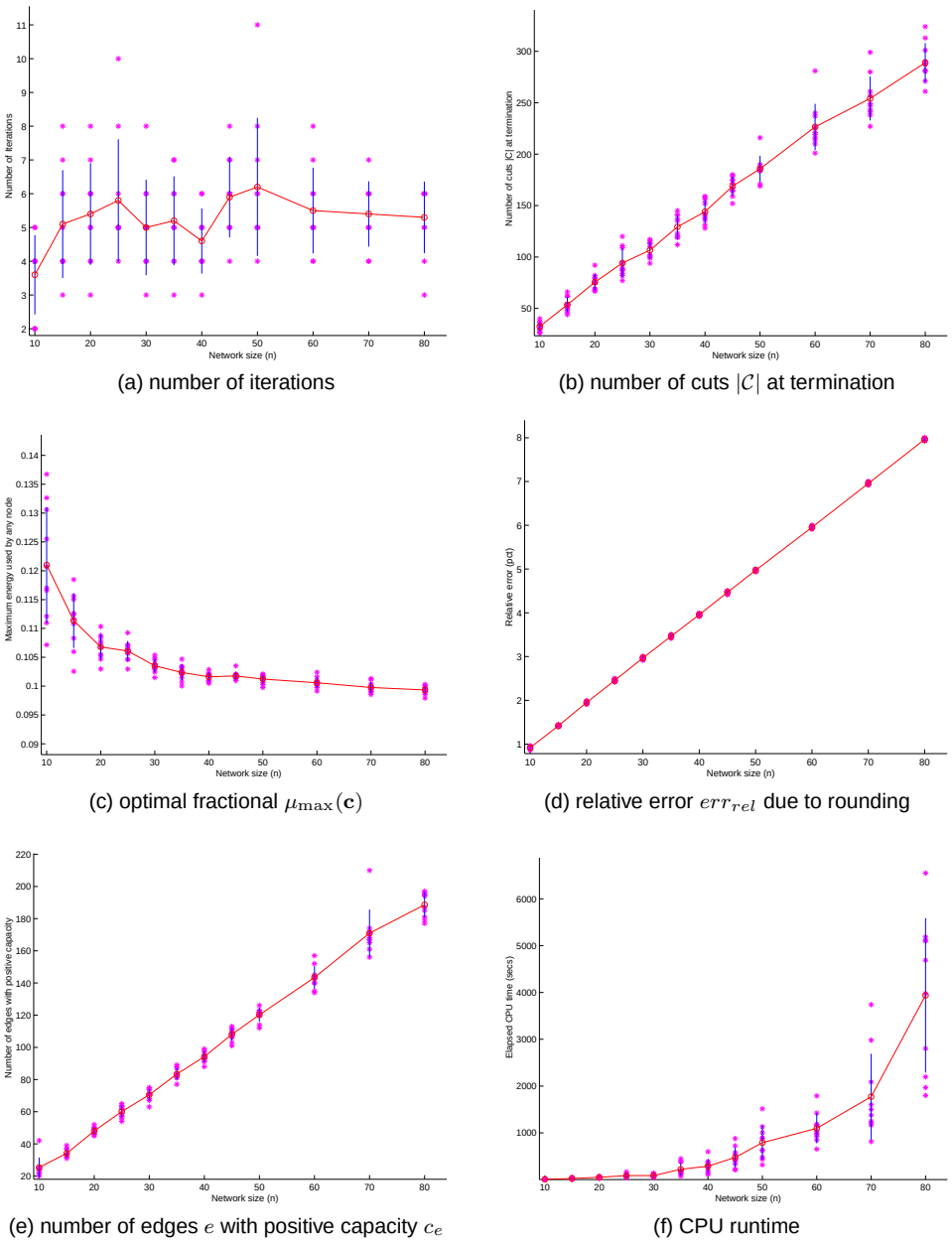
The average degree of the nodes is $|E^*_{\mathbf{c}}|/n \leq 2.5$, which is smaller than the bound of 4 provided by Theorem 4.7. Furthermore, as the network size increases, i.e. the sensor density increases, the maximum energy used by any sensor decreases. This is desirable and is due to the fact that the transmission energy costs decrease with increased density, while the receipt energy costs do not change significantly since the average degree of the sensors is ≤ 4.

Note from Fig. 2f that the variance of the CPU runtime increases as the the network size n increases. The ALGM^{2}EDA algorithm solves its largest linear program at the final iteration; that linear program has $|\mathcal{C}|$ constraints and $O(n^2)$ unknowns, and solving it takes time super–linear in $|\mathcal{C}|$. Consequently, since $|\mathcal{C}|$ increases as n increases, the variance of the CPU time increases super–linearly as n increases.

Finally, despite the rather slow PC used for the experiments and the Matlab interpretation overhead, problems of size up to 70 were solved on average in 30 minutes or less, and problems of size 80 were solved in about one hour of CPU time. This is a substantial improvement over the $\approx$ 2 hours and 1263 seconds average runtime to find optimal and approximately–optimal fractional solutions to the MLDA problem for networks with 60 nodes reported in [Kalpakis et al. 2003] and [Stanford and Tongngam 2006], respectively (see Table II. Note that the algorithms in [Kalpakis et al. 2003; Stanford and Tongngam 2006] have polynomial worst–case running times.

9. CONCLUSIONS

We considered two related problems for data gathering with in–network aggregation in wireless sensor networks. The MLDA problem is concerned with maximizing the system lifetime T so that we can perform T rounds of data gathering with in–network aggregation, given the initial available energy of the sensors. The M^{2}EDA problem is concerned with



(a) number of iterations

(b) number of cuts $|\mathcal{C}|$ at termination

(c) optimal fractional $\mu_{\max}(\mathbf{c})$

(d) relative error err_{rel} due to rounding

(e) number of edges e with positive capacity c_e

(f) CPU runtime

Fig. 2. Scatter plot together with the average and std. deviation of select performance metrics for the ALGM^{2}EDA algorithm run on 10 random M^{2}EDA instances, each with n nodes and demand function $\boldsymbol{\rho}_{b,T}$, where $T = 1000$.

Table II. Experimental average CPU running times (in secs) for finding near–optimal solutions to random instances of the MLDA problem with n nodes as reported in [Kalpakis et al. 2003, Table 6] and [Stanford and Tongngam 2006, Table 1]. [Stanford and Tongngam 2006] does not report any results on WSN with more than 60 nodes. For convenience, the average CPU running times of ALGM^{2}EDA are also shown.

n	ALGM^{2}EDA	MLDA [Kalpakis et al. 2003]	GK [Stanford and Tongngam 2006]
40	289	390	310
50	791	2880	643
60	1098	6510	1263
80	3938	15156	N/A

minimizing the maximum energy consumed by any sensor when performing T rounds of data gathering with in–network aggregation, for a given T.

We present the ALGM^{2}EDA algorithm for finding everywhere sparse, optimal within a factor α, integral solutions to the M^{2}EDA problem for a wireless sensor network with n nodes and lifetime requirement T, where $\alpha = 1 + 4n/T$. A solution is everywhere sparse if the number of communication links among the nodes in any subset X of nodes is $O(X)$, in our case at most $4|X|$ links. It follows that almost all sensors have small degree, and the overhead for each sensor for maintaining state information for data gathering with in–network aggregation is small. Further, with at most $4|X|$ communication links among any subset X of sensors, we expect less congestion and contention for the limited bandwidth almost everywhere. Since often $T = \omega(n)$, we obtain everywhere sparse asymptotically optimal integral solutions to the M^{2}EDA problem.

We also show that the MLDA and M^{2}EDA problems are essentially equivalent, in the sense that we can obtain an optimal fractional solution to an MLDA problem instance by scaling an optimal fractional solution to a suitable M^{2}EDA problem instance. As a result, we obtain everywhere sparse, asymptotically optimal integral solutions to the MLDA problem, when the initial available energy of the sensors is sufficient for supporting optimal fractional lifetime which is $\omega(n)$.

Extensive experimental results demonstrate the the ALGM^{2}EDA algorithm routinely finds optimal, everywhere sparse, fractional solutions to M^{2}EDA and MLDA problem instances with ≤ 70 nodes in less than 30 minutes CPU time. As part of our future work, we plan to investigate distributed algorithms for finding sparse near–optimal solutions to the M^{2}EDA and MLDA problems.

APPENDIX A. Linear Programming Primer

We provide an overview of some concepts in linear programming. The reader is referred to a text such as [Korte and Vygen 1991; Bertsimas and Tsitsiklis 1997] for further details.

Consider a linear program in standard form

$$\left\{ \begin{array}{ll} \min \mathbf{c}^T \mathbf{x} & \text{such that} \\ \\ \mathbf{A} \cdot \mathbf{x} = \mathbf{b} & \\ \\ \mathbf{x} \geq \mathbf{0} & \end{array} \right\} \tag{20}$$

where $\mathbf{A} \in \mathbb{R}^{n \times m}$, $\mathbf{c}, \mathbf{x} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^n$, and $n \leq m$. The linear program above defines a convex polyhedron $\mathcal{P} = \{\mathbf{x} \ : \ \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. For convenience, and without loss of generality, suppose that the constraint matrix $\mathbf{A}$ is of full–rank n and that $\mathbf{b} \geq \mathbf{0}$. The case where $\mathbf{A}$ has rank less than n leads to degeneracies requiring special handling,

see [Bertsimas and Tsitsiklis 1997]. We further assume, w.l.o.g., that the polyhedron $\mathcal{P}$ is bounded and non–empty, i.e. the linear program has a bounded optimal solution.

Let $\mathcal{B}$ be a sequence (ordered set) of n column indexes in $\{1, \ldots, m\}$. Let $\mathbf{A}_{\mathcal{B}}$ be the $n \times n$ sub–matrix of $\mathbf{A}$ whose ith column is $\mathbf{A}_{\mathcal{B}(i)}$. A sequence $\mathcal{B}$ is called a base if $\mathbf{A}_{\mathcal{B}}$ is of full-rank (invertible). It is called a feasible basis if $\mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b} \geq \mathbf{0}$. Since $\mathbf{A}$ is of full–rank and the linear program is feasible, a feasible basis always exists. A variable x_i (column $\mathbf{A}_i$) with index in $\mathcal{B}$ is called a *basic variable (basic column)*, otherwise it is called a *non–basic variable (non–basic column)*.

Construct a feasible solution $\mathbf{x}$ corresponding to a feasible base $\mathcal{B}$ by taking $\mathbf{x}_{\mathcal{B}} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$ and $\mathbf{x}_{\overline{\mathcal{B}}} = \mathbf{0}$. Such a solution is called a *basic feasible solution* (bfs). There is a bijection between basic feasible solutions and vertices (extreme points) of the polytope defined by $\mathbf{A}$. Furthermore, an optimal solution always occurs at one of its vertices.

Associate with each constraint a *shadow price* (or *dual variable*). The shadow prices $\boldsymbol{\pi} \in \mathbb{R}^n$ corresponding to a base $\mathcal{B}$ is given by

$$\boldsymbol{\pi}^T = \mathbf{c}_{\mathcal{B}}\mathbf{A}_{\mathcal{B}}^{-1}. \tag{21}$$

The *relative cost* $\bar{c}_j$ of each non–basic column $\mathbf{A}_j$ is given by

$$\bar{c}_j = c_j - \boldsymbol{\pi}^T\mathbf{A}_j \tag{22}$$

The Simplex method, discovered by Dantzig, systematically explores the set of basic feasible solutions, starting from an initial bfs, until an optimal bfs is found. The process of moving from a bfs to an adjacent bfs is called *pivoting*. In pivoting, we exchange a basic column with a non–basic column, without increasing the cost of the best feasible solution so far.

We describe next a variant of the Simplex method, the Revised Simplex Method (RSM) with the lexico–min rule. An arbitrary non–basic column $\mathbf{A}_j$ enters the current base $\mathcal{B}$ if its relative cost $\bar{c}_j < 0$. If all non–basic columns have relative cost ≥ 0, then the current bfs is optimal and Simplex terminates. Otherwise, a basic column to exit the current base $\mathcal{B}$ needs to be selected. There are multiple approaches to do so. We describe the lexico–min approach for choosing the basic column to exit the current basis $\mathcal{B}$, since it guarantees termination in a finite number of pivoting steps [Bertsimas and Tsitsiklis 1997]. Let $\mathbf{a}_i$ denote the ith row of the matrix $\mathbf{A}_{\mathcal{B}}$. Let l be the index of the lexicographically smallest row $[x_i, \mathbf{a}_i]/u_i$ with $u_i > 0$,

$$l = \arg \text{lexico–min} \left\{ \frac{[x_i, \mathbf{a}_i]}{u_i} \ : \ u_i > 0 \right\}, \tag{23}$$

where $\mathbf{u} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{A}_j$ and $\mathbf{x} = \mathbf{A}_{\mathcal{B}}^{-1}\mathbf{b}$. Column $\mathbf{A}_j$ enters the base $\mathcal{B}$ replacing column $\mathbf{A}_{\mathcal{B}(l)}$, i.e. $\mathcal{B}(l) \longleftarrow j$. An index l always exists, since otherwise $u_i \leq 0$ for all i and the problem is unbounded.

Extensive computational experience since the discovery of the Simplex method demonstrated that in practice it is an efficient algorithm. The Revised Simplex method offers computational advantages for linear programs with sparse constraint matrices. Moreover, observe that RSM allows us to solve linear programs with exponentially many variables by performing few pivots in practice, provided that we can either find, in polynomial–time, a non–basic column with negative relative cost or show that no such column exists.

There are many cases, where other methods, such as interior–point methods, run faster. For example, the standard solver in Matlab is based on interior–point methods. Optimal

solutions computed by interior–point methods may not be optimal basic solutions, i.e. may have support larger than n. Obtaining optimal basic solutions from a pair of non–basic optimal solutions is known as basis crushing, purification, or crossover. Beling and Megiddo [Beling and Megiddo 1998] describe efficient algorithms for basis crushing for linear programs with polynomially many constraints and unknowns.

In our experiments of solving instances $\langle G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\rho} \rangle$ of the M^{2}EDA problem we do the following. At each iteration, we first find an optimal non-basic feasible solution $\mathbf{x}$ to the linear program $\langle \min \mu_{\max}(\mathbf{x})$ such that $\mathbf{x}(S) \geq \rho(S),\ \forall S \in \mathcal{C} \rangle$, where $\mathcal{C}$ is the current collection of directed cuts. Then, we do basis crashing on $\mathbf{x}$ by by solving this linear program again using the Revised Simplex method, but this time we consider only edges in the subgraph $G^*_{\mathbf{x}}$ instead of G.

ACKNOWLEDGMENTS

We would like to thank the referees for their valuable comments and suggestions for improving this paper.

REFERENCES

AKYILDIZ, I. F., SU, W., SANKARASUBERMANIAN, Y., AND CAYIRICI, E. 2002. A survey of sensor networks. *IEEE Communications Magazine 40*, 102–114.

BELING, P. A. AND MEGIDDO, N. 1998. Using fast matrix multiplication to find basic solutions. *Theoretical Computer Science 205*, 307–316.

BERTSIMAS, D. AND TSITSIKLIS, J. N. 1997. *Introduction to Linear Optimization.* Athena Scientific.

BHARDWAJ, M., GARNETT, T., AND CHANDRAKASAN, A. 2001. Upper bounds on the lifetime of sensor networks. In *Proc. of International Conference on Communications*.

CHANG, J.-H. AND TASSIULAS, L. 2004. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking 12*, 4, 609–619.

FRANK, A. 1993. Applications of submodular functions. In *Surveys in Combinatorics, London Mathematical Society Lecture Note Series 187*, K. Walker, Ed. Cambridge University Press, 85–136.

GARG, N. AND KONEMANN, J. 1998. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*. 300–309.

GOEMANS, M. X. 2006. Minimum bounded degree spanning trees. In *FOCS*. IEEE Computer Society, 273–282.

GRÖTSCHEL, M., LOVÁSZ, L., AND SCHRIJVER, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica 1*, 169–197.

HEINZELMAN, W., KULIK, J., AND BALAKRISHNAN, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. of 5th ACM/IEEE Mobicom Conference*.

HEINZELMAN, W. R., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*.

HOU, Y. T., SHI, Y., PAN, J., AND MIDKIFF, S. F. 2006. Maximizing the lifetime of wireless sensor networks through optimal single-session flow routing. *IEEE Transactions on Mobile Computing 5*, 9, 1255–1266.

HURKENS, C. A. J., LOVASZ, L., SCHRIJVER, A., AND TARDOS, E. 1988. How to tidy up your set–system? *Combinatorics (A. Hajnal, L. Lovsz and V.T. Ss, eds.)*, 309–314.

JAIN, K. 2001. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica 21*, 1, 39–60.

KAHN, J. M., KATZ, R. H., AND PISTER, K. S. J. 1999. Mobile networking for smart dust. In *Proc. of 5th ACM/IEEE Mobicom Conference*.

KALPAKIS, K., DASGUPTA, K., AND NAMJOSHI, P. 2003. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks 42*, 6, 697–716.

KORTE, B. AND VYGEN, J. 1991. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag.

KRISHNAMACHARI, B., ESTRIN, D., AND WICKER, S. 2002. The impact of data aggregation in wireless sensor networks. In *Proc. of International Workshop on Distributed Event-Based Systems*.

LINDSEY, S. AND RAGHAVENDRA, C. S. 2002. PEGASIS: Power efficient gathering in sensor information systems. In *Proc. of IEEE Aerospace Conference*.

LINDSEY, S., RAGHAVENDRA, C. S., AND SIVALINGAM, K. 2001. Data gathering in sensor networks using the energy*delay metric. In *Proc. of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing*.

LIU, X., HUANG, Q., AND ZHANG, Y. 2004. Combs, needles, haystacks: Balancing push and pull for discovery in largescale sensor networks. In *ACM Sensys*. 122–133.

LUO, J. AND HUBAUX, J.-P. 2005. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *Proc. of INFOCOM*. 1735–1746.

MADDEN, S., FRANKLIN, M. J., HELLERSTEIN, J. M., AND HONG, W. 2002. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proc. of 5th Symposium on Operating Systems Design and Implementation*. 131–.

MADDEN, S., SZEWCZYK, R., FRANKLIN, M. J., AND CULLER, D. 2002. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *Proc. of 4th IEEE Workshop on Mobile Computing and Systems Applications*.

MIN, R., BHARDWAJ, M., CHO, S., SINHA, A., SHIH, E., WANG, A., AND CHANDRAKASAN, A. 2001. Low-power wireless sensor networks. In *VLSI Design*.

PAN, J., CAI, L., HOU, Y. T., SHI, Y., AND SHEN, S. X. 2005. Optimal base-station locations in two-tiered wireless sensor networks. *IEEE Transactions on Mobile Computing 4,* 5, 458–473.

PUTTAGUNTA, V. AND KALPAKIS, K. 2007. Accuracy vs. lifetime: Linear sketches for aggregate queries in sensor networks. *Algorithmica 49,* 4, 357–385.

RABAEY, J., AMMER, J., DA SILVA JR, J., AND PATEL, D. 2000. PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes. In *Proc. of the IEEE Computer Society Annual Workshop on VLSI*.

SANKAR, A. AND LIU, Z. 2004. Maximum lifetime routing in wireless ad-hoc networks. In *Proc. of INFOCOM*.

STANFORD, J. AND TONGNGAM, S. 2006. Approximation algorithm for maximum lifetime in wireless sensor networks with data aggregation. In *Proc. of SNPD*. Washington, DC, USA, 273–277.

YU, Y., PRASANNA, V. K., AND KRISHNAMACHARI, B. 2006. Energy minimization for real-time data gathering in wireless sensor networks. *IEEE Transactions on Wireless Communications 5,* 11, 3087–3096.

Received August 2008; revised May 2009, December 2010; accepted January 2010.