A Combinatorial Algorithm for the Maximum Lifetime Data Gathering and Aggregation Problem in Sensor Networks

Konstantinos Kalpakis and Shilang Tang
Computer Science & Electrical Engineering Department
University of Maryland Baltimore County
{kalpakis, stang2}@csee.umbc.edu

***Abstract*—The Maximum Lifetime Data Gathering and Aggregation (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. A solution of value T to the MLDA problem consists of a collection of aggregation trees together with the number of rounds each such tree should be used in order to achieve lifetime T.**

We describe a combinatorial iterative algorithm for finding an optimal continuous solution to the MLDA problem that consists of up to $n-1$ aggregation trees and achieves lifetime T_o, which depends on the network topology and initial energy available at the sensors. We obtain an α–approximate optimal integral solution by simply rounding down the optimal continuous solution, where $\alpha = (T_o - n + 1)/T_o$. Since in practice $T_o \gg n$, $\alpha \approx 1$. We get asymptotically optimal integral solutions to the MLDA problem whenever the optimal continuous solution is $\omega(n)$. Furthermore, we demonstrate the efficiency and effectiveness of the proposed algorithm via extensive experimental results.

I. Introduction

Recent technological advances have led to the development of wireless networks of micro–sensors [7], [10], [11]. Such networks are expected to consist of numerous inexpensive micro–sensors, readily deployable in various physical environments to collect useful information (e.g. seismic, acoustic, medical and surveillance data) in a robust and autonomous manner. However, there are several obstacles that need to be overcome before this vision becomes a reality – see Akyildiz et al [1] for a comprehensive survey of issues arising in wireless sensor networks. 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. Conserving the sensor energy, so as to maximize the system's lifetime — the time until the first sensor depletes its energy, has been one of the key challenges.

In this paper we consider the problem of Maximum Lifetime Data Gathering with in-network Aggregation (MLDA) in wireless sensor networks. We adopt the framework introduced by Kalpakis et al [8] and Dasgupta et al [5]. In [8], the notion of data aggregation tree is introduced, which is a directed tree in which every sensor has a directed path towards the base station. The maximum lifetime data gathering and aggregation problem is essentially reduced in [8] to a directed network design problem: maximize T such that all sensor–base station directed cuts have capacity at least T while the total energy consumed by sending/receiving messages along the edges incident to each sensor does not exceed each sensor's available energy. Even though the network design problem is NP–hard, heuristics in [8] show that (1) tight approximate solutions to the network design problem can be obtained in reasonable time for small to medium–size networks, and (2) a collection of aggregation trees, together with the number of rounds each such tree is to be used in order to achieve lifetime T, can be determined in polynomial–time.

Dasgupta et al [5] consider an equivalent formulation of the MLDA problem and provide approximate solutions by selecting aggregation trees from a pool of randomly constructed aggregation trees $\mathcal{T}$, and then using linear programming to determine the number of rounds each tree in the pool should be used in order to maximize the lifetime.

We provide a combinatorial iterative algorithm for the MLDA problem based on the formulation in [5], while implicitly considering all the n^{n-2} aggregation trees for a WSN with n nodes. Our algorithm is based on the Revised Simplex Method with a column generation scheme. In particular, the original contributions of this paper is as follows:

- provide a combinatorial iterative algorithm (called RSM–MLDA) for finding an optimal continuous solution to the MLDA problem, which consists of at most $n - 1$ aggregation trees.
- find α–approximate integral solutions to the MLDA problem for a WSN with n nodes and optimal continuous lifetime T_o, where $\alpha = (T_o - n + 1)/T_o$. Such solutions are computed from rounding down solutions obtained by the RSM–MLDA algorithm.
- provide an upper bound on the system lifetime, which is easily computable during each iteration of the RSM–MLDA algorithm. This upper bound enables us to estimate a lower bound on the approximate ratio of the solution at each iteration of RSM–MLDA, and hence terminate early if that ratio exceeds a desired threshold.

Moreover, we conduct an extensive experimental evaluation of

the proposed approach illuminating its practicability.

A. Related work

Stanford and Tongngam [13] give an $(1-\epsilon)^2$–approximation iterative algorithm, based on the Garg–Konemann approach [6] for solving packing linear programs, where the linear program is the same LP formulation of the MLDA problem considered here. Their approximation algorithm inherits the aspects of Garg–Konemann approach. A markable limitation of Garg–Konemann approach is the slow convergence for small ϵ due to the large number of iterations. A large number of iterations typically leads to a drastic slowdown of their algorithm. Moreover, it does not guarantee a small number of aggregation trees – a property that is critical in WSNs since the overhead of distributing information about these trees to the sensors must be limited. In addition, the integral solution obtained by rounding down the achieved continuous solution will have substantially reduced lifetime.

Xue at al [15] approach the MLDA problem as a maximum concurrent multicommodity flow problem in digraphs. Their tree-based $(1-\epsilon)^2$–approximation algorithm is also based on the Garg–Konemann approach [6] therefore it inherits its performance aspects as well. Furthermore, it turns out, as Stanford and Tongngam [13] point out, that the model used by Xue at al [15] does not allow the same manner of data aggregation we consider here.

The rest of the paper is organized as follows. In section II we give the necessary preliminaries, together with a brief overview of linear programming in section III. We describe our combinatorial algorithm for the MLDA problem in section IV, with an algorithm for finding minimum weight branchings in section IV-B. An upper bound on the optimal solution of the MLDA problem is given in section IV-C. Experimental results are provided in section V, and we conclude in section VI.

II. Preliminaries

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 assumed to have limited non-replenishable 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 so that they become available at the base station. During data gathering, in–network aggregation is assumed, i.e. any number of incoming data packets at a node can be aggregated into a single outgoing data packet. 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) $G=(V,E)$, with $V=\{1,2,\ldots,n\}$ and $E\subseteq V\times V$. There exists an edge $ij\in E$ whenever i can successfully send a packet to j. Let d_{ij} be the distance between i and 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. Let ϵ_i be the energy available at node i. We assume, w.lo.o.g, that $\epsilon_b=\infty$, $r_b=0$, $\tau_{bi}=0$, $\forall i\in V$. For simplicity we consider sensor networks with a single base station. Sensor networks with multiple base stations can be easily handled as follows. Introduce a new node, b', to serve as the new single base station, and then append to G, for each current base station i, the new edge ib' with $\tau_{ib'}=0$.

Given a digraph $G=(V,E)$, a *branching* $\mathcal{T}$ rooted at node $b\in V$ is a subgraph of G such that each node has a directed path to b in $\mathcal{T}$, and its undirected version is a tree spanning all the nodes in V. In the context of WSNs, we refer to such a branching $\mathcal{T}$ as an *aggregation tree* since it indicates how to gather data from all the sensors to the base station with in–network aggregation.

We denote vectors and matrices with lower and upper case bold letters, e.g. $\mathbf{x}$ and $\mathbf{A}$ is a vector and matrix respectively. 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).

Given an instance $\mathfrak{I}$ of an 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 $\textsc{sol}(\mathfrak{I})/\textsc{opt}(\mathfrak{I})$. We refer to continuous (integral) solutions to instances of optimization problems, whenever fractional values for their unknowns are allowed (not allowed).

Finally, let us comment on the work by Garg and Konemann [6]. Garg and Konemann [6] describe an important approach to obtain simple iterative algorithms with provable approximation ratios for the maximum multicommodity flow, packing linear program, maximum concurrent flow, and minimum cost multicommodity flow problems. These algorithms utilize a parameter ϵ, and at each iteration compute a variable of least relative cost whose value is incremented by a certain small amount. For the maximum multicommodity flow and the packing LP problems they achieve an $(1-\epsilon)^2$ approximation ratio with at most $km\lceil\log_{1+\epsilon}m/\epsilon\rceil$ iterations, where m is the number of edges and constraints and k is the number of commodities or 1, respectively. For the maximum concurrent flow and minimum cost multicommodity flow problems they achieve an $(1-\epsilon)^3$ approximation ratio with at most $3k\lg k\lceil\log_{1+\epsilon}(\frac{m}{1-\epsilon})/\epsilon$ iterations, where m is the number of edges and k is the number of commodities. Note that, In all four problems, the number of iterations needed to achieve an approximation ratio of $\alpha\leq 1$ is $O([(1-\sqrt[K]{\alpha})\log(2-\sqrt[K]{\alpha})]^{-1})$ for fixed problem size, with $K=2$ or 3. Further, when applied to solve packing linear programs, it finds solutions whose support can be as high as the number of iterations. Nevertheless, the Garg–Konemann approach is attractive since it takes considerable time to solve such problem using traditional linear programming algorithms as the size of the problem increases, and this beautiful ap-

proach has been applied to various problems in WSNs, e.g. to the maximum lifetime routing problem [4] and to the problem of maximizing data collection in store–and-extract networks [12].

III. A Linear Programming Primer

We provide an overview of some concepts in linear programming we will use later on. The reader is referred to a text such as [3], [9] for further details.

Consider a linear program in standard form

$$\left\{ \begin{array}{ll} \min \mathbf{c}^T \cdot \mathbf{x} & \text{such that} \\ \\ \mathbf{A} \cdot \mathbf{x} = \mathbf{b}, & \\ \\ \mathbf{x} \geq \mathbf{0}, & \end{array} \right\} \tag{1}$$

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

$$\mathcal{P} = \{\mathbf{x} \ : \ \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}. \tag{2}$$

For convenience and w.l.o.g., we assume that the constraint matrix $\mathbf{A}$ is of full–rank n and that $\mathbf{b} \geq 0$. The case where $\mathbf{A}$ has rank less than n leads to degeneracies requiring special handling, see [3]. 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 $\mathbf{A}_{\mathcal{B}(i)}$. The sequence $\mathcal{B}$ is called a *basis* if $\mathbf{A}_\mathcal{B}$ is of full-rank (invertible). It is called a *feasible basis* if $\mathbf{A}_\mathcal{B}^{-1} \cdot \mathbf{b} \geq \mathbf{0}$. Since $\mathbf{A}$ is of full–rank and the linear program is feasible, a feasible basis always exists. A variable x_i (column $\mathbf{A}_i$) whose index is in $\mathcal{B}$ is called a *basic* variable (column), otherwise it is called a *non–basic* variable (column).

Construct a feasible solution $\mathbf{x}$ corresponding to a feasible basis $\mathcal{B}$ by taking $\mathbf{x}_\mathcal{B} = \mathbf{A}_\mathcal{B}^{-1} \cdot \mathbf{b}$ and $\mathbf{x}_{\overline{\mathcal{B}}} = \mathbf{0}$. Such a solution is called a *basic feasible solution* (bfs). The support $\mathbb{I}(\mathbf{x})$ of any bfs $\mathbf{x}$ has size at most n. There is a bijection between basic feasible solutions and extreme points (vertices) of the polyhedron $\mathcal{P}$. Furthermore, an optimal solution always occurs at one of $\mathcal{P}$'s vertices.

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

$$\boldsymbol{\pi}^T = \mathbf{c}_\mathcal{B} \cdot \mathbf{A}_\mathcal{B}^{-1}. \tag{3}$$

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

$$\bar{c}_j = c_j - \boldsymbol{\pi}^T \cdot \mathbf{A}_j \tag{4}$$

The Simplex method, discovered by Dantzig, systematically explores the extreme points (bfs) of $\mathcal{P}$ starting from an initial extreme point, until an optimal extreme point is found. The process of moving from an extreme point to an adjacent extreme point is called *pivoting*. In pivoting, we move from an extreme point to an adjacent extreme point with smaller or equal cost, by exchanging a basic column with a non–basic column.

Next we describe a variant of the Simplex method, the Revised Simplex Method (RSM) with the lexico–min rule. An arbitrary non–basic column $\mathbf{A}_j$ can enter the current basis $\mathcal{B}$ if its relative cost $\bar{c}_j < 0$. If all non–basic columns have relative cost ≥ 0, then the current bfs is optimal and the algorithm terminates. Otherwise, a basic column to exit the current basis $\mathcal{B}$ is selected. There are multiple approaches to do that. We describe the lexico–min rule for choosing a basic column that exits the current basis $\mathcal{B}$, since it guanrantees termination in a finite number of pivoting steps [3]. Let $\mathbf{a}_i$ denote the ith row of $\mathbf{A}_\mathcal{B}$. Let l be the index of the lexicographically smallest row $\mathbf{a}_i / u_i$ with $u_i > 0$,

$$l = \arg \text{lexico–min} \left\{ \frac{\mathbf{a}_i}{u_i} \ : \ u_i > 0 \right\}, \tag{5}$$

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

Extensive computational experience since Simplex's discovery demonstrated that in practice it is an efficient algorithm [3]. RSM offers computational advantages for sparse linear programs. Moreover, RSM facilitates the solution of linear programs with exponentially many variables by performing few pivots in practice, provided that we can efficiently either find a non–basic column with negative relative cost or show that no such column exists.

IV. A Combinatorial Algorithm for the MLDA problem

In this section we provide our combinatorial algorithm for the MLDA problem with the same LP formulation used in [5], while considering all the n^{n-2} aggregation trees of a WSN with n nodes. Our algorithm is based on the Revised Simplex method with the lexico–min rule.

Consider an instance $\mathfrak{I} = \langle\, G = (V, E), b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem. Let $\mathcal{T}$ be the set of all aggregation trees (branchings rooted at b) for the instance $\mathfrak{I}$. A feasible solution to $\mathfrak{I}$ can be described by a collection of aggregation trees $T_i \in \mathcal{T}$ together with the number of rounds x_i each tree is used. There is a natural bijection between the branchings of $G = (V, E)$ which are rooted at b and the $|V|^{|V|-2}$ labelled trees spanning these nodes: direct all edges of a labelled tree towards b to a get the correspondig branching. Hence, the size of $\mathcal{T}$ is $|V|^{|V|-2}$. Let $\mu_j(T_i)$ be the energy consumed by node j when performing data gathering with in–network aggregation using the branching $T_i \in \mathcal{T}$.

A continuous (integral) solution to the MLDA problem instance $\mathfrak{I}$ can be found by solving the following linear (mixed

integer) program in standard form

$$\text{LP:} \left\{ \begin{array}{ll} \max \sum_{T_i \in \mathcal{T}} x_i & \text{such that} \\ s_j + \sum_{T_i \in \mathcal{T}} x_i \cdot \mu_j(T_i) = \epsilon_j, & \forall j \in V \\ x_i \geq 0, & \forall T_i \in \mathcal{T} \\ s_j \geq 0, & \forall j \in V \end{array} \right\} \quad (6)$$

Extend $\mathbf{x}$ with n components, such that x_{m+i} is identified with the ith slack variable s_i, where $m = |\mathcal{T}|$. Define the $n \times (m+n)$ matrix $\mathbf{A}$ so that $\mathbf{A}_i = \mu(T_i)$, $i = 1, 2, \ldots, m$, and $\mathbf{A}_{m+i}$ is the ith column of the $n \times n$ identity matrix $\mathbf{I}_n$, $i = 1, 2, \ldots, n$. Let $\mathbf{c} \in \mathbb{R}^{m+n}$ have its first m components equal to -1, and with all the rest equal to 0. Let $\mathbf{b} \in \mathbb{R}^n$ be equal to the vector of initial energies $\boldsymbol{\epsilon}$ of the sensors. The LP above is now in the standard form

$$\text{LP:} \left\{ \begin{array}{ll} \min \mathbf{c}^T \cdot \mathbf{x} & \text{such that} \\ \mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \\ \mathbf{x} \geq \mathbf{0}, \end{array} \right\} \quad (7)$$

An initial feasible basis for LP is $\mathcal{B} = \langle\, m+1, m+2, \ldots, m+n \,\rangle$, with corresponding bfs $\mathbf{x}_\mathcal{B} = \mathbf{b} = \boldsymbol{\epsilon}$. The value $\mathbf{c}^T \cdot \mathbf{x}$ of each feasible solution $\mathbf{x}$ is the achieved system lifetime. In the LP above, we explicitly store up to n columns of $\mathbf{A}$ at any point in time. All other columns of $\mathbf{A}$ will be generated on demand as needed.

During each pivoting step of RSM for the LP, the shadow prices vector is $\boldsymbol{\pi}^T = \mathbf{c}_\mathcal{B} \cdot \mathbf{A}_\mathcal{B}^{-1}$. Let us define the *unit energy price* at a node j as $\phi_j = -\pi_j$, and the energy cost of a branching $T_k \in \mathcal{T}$ to be equal to $\sum_{j \in V} \phi_j \mu_j(T_k)$. Observe that the energy cost of T_k is equal to

$$\sum_{j \in V} \phi_j \cdot \mu_j(T_k) = \sum_{j \in V} \phi_j \left(\sum_{ji \in T_k} \tau_{ji} + \sum_{ij \in T_k} r_j \right) = \sum_{ij \in T_k} (\phi_i \tau_{ij} + \phi_j r_j) = \sum_{ij \in T_k} w_{ij} = w(T_k), \quad (8)$$

where we define the weight of an edge $ij \in E$ to be equal to $w_{ij} = \phi_i \tau_{ij} + \phi_j r_j$.

Now consider a non–basic column $\mathbf{A}_k$ corresponding to a branching $T_k \in \mathcal{T}$. Its relative cost is

$$\bar{c}_k = c_k - \boldsymbol{\pi}^T \cdot \mu(T_k) = -1 - \boldsymbol{\pi}^T \cdot \mu(T_k) = w(T_k) - 1. \quad (9)$$

Note that T_k is a candidate to enter the current basis $\mathcal{B}$ if its energy cost $w(T_k)$ is less than 1 [1]. Therefore, it is sufficient to find a branching $T_k \in \mathcal{T}$ with minimum energy cost, and if its cost is < 1, then T_k enters the basis $\mathcal{B}$, otherwise, the pivoting terminates. Upon finding such a branching T_k, the variable x_k enters the basis $\mathcal{B}$, and a basic column $\mathbf{A}_l$ is chosen to leave $\mathcal{B}$ using the lexico–min rule. The basic column $\mathbf{A}_l$ corresponds to the branching $T_{\mathcal{B}(l)}$, if $\mathcal{B}(l) \leq m$, or to the slack variable $s_{\mathcal{B}(l)-m}$ otherwise [2].

Algorithm 1 The RSM–MLDA algorithm for finding an optimal continuous solution to the MLDA problem.

Input: MLDA instance $\mathfrak{I} = \langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$
Output: maximum lifetime T, together with at most $|V| - 1$ branchings rooted at b and their #rounds

1. $\mathbf{D} \longleftarrow$ identity matrix of order $|V|$
2. let $\mathbf{d}_i$ denote the ith row of $\mathbf{D}$
3. **for** each $1 \leq i \leq |V|$ **do**
4. $\quad branching[i] \longleftarrow$ **null**
5. $\quad rounds[i] \longleftarrow 0$
6. $\quad c[i] \longleftarrow 0$
7. **for** at most $|V|^{|V-2|}$ iterations **do**
8. $\quad \phi \longleftarrow -\mathbf{c} \cdot \mathbf{D}^{-1}$
9. $\quad$ **for** each edge $ij \in E$ **do**
10. $\quad\quad w_{ij} \longleftarrow \phi_i \tau_{ij} + \phi_j r_j$
    // find candidate branchings to enter and leave the current basis
11. $\quad T_k \longleftarrow$ GetMWB(G, b, w)
12. $\quad$ **if** $w(T_k) \geq 1$ **then**
13. $\quad\quad$ **break**
14. $\quad \mathbf{u} \longleftarrow \mathbf{D}^{-1} \cdot \mu(T_k)$
15. $\quad$ **if** the support $\mathbb{I}(\mathbf{u})$ is empty **then**
16. $\quad\quad$ **return** *instance is unbounded*
17. $\quad l \longleftarrow \arg$ lexico–min$\{\mathbf{d}_i / u_i \ : \ i \in \mathbb{I}(\mathbf{u})\}$
18. $\quad branching[l] \longleftarrow T_k$
19. $\quad D_l \longleftarrow \mu(T_k)$
20. $\quad c[l] \longleftarrow -1$
    // update the rounds for each branching in the basis
21. $\quad rounds \longleftarrow \mathbf{D}^{-1} \boldsymbol{\epsilon}$
22. $\quad$ **for** $1 \leq i \leq |V|$ and $branching[i] =$ **null** **do**
23. $\quad\quad rounds[i] \longleftarrow 0$
24. $T \longleftarrow$ sum of all $rounds[i]$
25. **return** $\langle\, T, rounds, branching \,\rangle$

The complete description of our iterative algorithm for finding a continuous optimal solution to the MLDA problem is given in Algorithm 1. Our algorithm finds mimimum weight branchings in a digraph using an algorithm given in section IV-B below. The correctness of our algorithm follows from the discussion above and the properties of the Revised Simplex method with the lexico–min rule. The memory requirements of our algorithm is $O(V^2)$. The worst–case running time of each iteration of our algorithm is bounded as follows: $O(VE+V^2)$ for finding a minimum weight branching, $O(V^3)$ time to invert the matrix $\mathbf{D}$, and $O(V^2)$ total time for all other operations, for a grand total of $O(V^3)$ worst–case running time per iteration. The running time can be reduced to $O(VE+V^2)$ by utilizing the matrix–inversion lemma to update $\mathbf{D}^{-1}$ after

[1] The energy cost of the branching that corresponds to a basic column is equal to 1.

[2] The slack variable x_{m+b} never exits the basis, since $\epsilon_b > 0$ and $\mu_b(T_i) = 0$ for all $T_i \in \mathcal{T}$.

each pivoting step in $O(V^2)$ time instead of recomputing $\mathbf{D}^{-1}$ each time from scratch.

A. Getting α–approximate integral solutions

Consider an instance $\mathfrak{I} = \langle\, G = (V, E), b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem and a continuous optimal basic solution $\mathbf{x}$ with lifetime T, computed by our RSM–MLDA algorithm. Since the support of $\mathbf{x}$ has size at most $|V|$ and the slack variable that corresponds to b is always basic, the number of distinct aggregation trees in the continuous optimal bfs is $\leq |V|-1$. By rounding down the components of $\mathbf{x}$, we get an integral solution to $\mathfrak{I}$ with lifetime $\geq T-|V|+1$, i.e. we get an integral solution to MLDA which is optimal within a factor of $\alpha = (T-|V|+1)/T$. For example, if $T = 1000$ and $n = 50$, we get an integral solution which is optimal within a factor of 95%. Whenever $T = \Theta(\boldsymbol{\epsilon}) = \omega(n)$, we find asymptotically optimal integral solutions to the MLDA problem.

B. Computing Minimum Weight Branchings

Algorithm 2 The GETMWB algorithm for finding a minimum weight branching rooted at r.

Input: $\langle\, G = (V, E), r, \mathbf{w} \,\rangle$
Output: minimum weight branching $\mathcal{T}$ rooted at r

1. let $E_o = \{\ e \in E\ :\ w_e = 0\}$
2. let $G_o = (V, E_o)$
3. **while** at most $|V|$ iterations **do**
4. let T_k be the transpose of a BFS tree of G_o^T rooted at r
5. **if** T_k spans all the nodes in V **then**
6. **return** T_k
7. **for** each SCC $G_i = (V_i, E_i)$ of G_o **do**
8. $\partial \longleftarrow E \cap V_i \times \overline{V_i}$
9. $\theta \longleftarrow \min\{w_e\ :\ e \in \partial\}$
10. **for** each $e \in \partial$ **do**
11. $w_e \longleftarrow w_e - \theta$
12. **return** no branching exists

We turn our attention to finding a minimum weight branching for a digraph $G = (V, E)$ rooted at a node $r \in V$ and with edge weights $w : E \longrightarrow \mathbb{R}$. Tarjan [14] provides an more efficient algorithm for the minimum weight branching for a digraph $G = (V, E)$ whose running time is $O(\min\{E \log V, V^2\})$. We provide a simple, but slower, algorithm whose detailed description is given in Algorithm 2. The algorithm is an instance of the local–ratio paradigm [2]. Correctness of the algorithm can be shown by induction on the number of iterations of the outer–loop, and using the following observation. For each iteration and each strongly connected component (SCC) G_i found during that iteration, any minimum weight branching must use an edge leaving each such G_i that does not contain the root. Since each iteration of the outer–loop takes $O(V + E)$ time, and the number of iterations is at most $|V|$, the worst–case running time of GETMWB is $O(V^2 + VE) = O(V^3)$. Its total memory requirements is $O(V^2)$.

C. An upper bound on the lifetime

We compute an upper bound on the lifetime during each iteration of the RSM–MLDA algorithm using duality theory of linear programming. Consider an instance $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ of the MLDA problem, a convenient way to write a primal linear program equivalent to the one in (6) is

$$\left\{ \begin{array}{ll} \max \mathbf{1}^T \cdot \mathbf{x} & \text{such that} \\ \mathbf{A} \cdot \mathbf{x} \leq \boldsymbol{\epsilon}, & \\ \mathbf{x} \geq \mathbf{0}, & \mathbf{x} \in \mathbb{R}^{|\mathcal{T}|} \end{array} \right\} \tag{10}$$

where the matrix $\mathbf{A}$ has as columns the vectors $\mu(T_k)$, for all branching $T_k \in \mathcal{T}$, and $\mathcal{T}$ is the set of all branchings of G rooted at b. The dual of this linear program is

$$\left\{ \begin{array}{ll} \min \boldsymbol{\epsilon}^T \cdot \mathbf{y} & \text{such that} \\ \mathbf{A}^T \cdot \mathbf{y} \geq \mathbf{1}, & \\ \mathbf{y} \geq \mathbf{0}, & \mathbf{y} \in \mathbb{R}^{|V|} \end{array} \right\} \tag{11}$$

Observe that both primal and dual linear programs above are feasible. Consider a pair of feasible solutions $\mathbf{x}$ and $\mathbf{y}$ to the linear programs in (10) and (11). Strong duality implies that

$$\mathbf{1}^T \cdot \mathbf{x} \leq \boldsymbol{\epsilon}^T \cdot \mathbf{y} \tag{12}$$

with equality when both $\mathbf{x}$ and $\mathbf{y}$ are respectively optimal.

Algorithm 3 The **GetUB** procedure for computing an upper bound on the system lifetime.

Input: MLDA instance $\langle\, G, b, \boldsymbol{\tau}, \mathbf{r}, \boldsymbol{\epsilon} \,\rangle$ and energy prices $\boldsymbol{\phi}$
Output: upper bound on the system lifetime.

1. $\mathbf{z} \longleftarrow \max\{-\boldsymbol{\phi}_{V-b}, \mathbf{0}\}$
2. **for** each edge $ij \in E$ **do**
3. $w_{ij} \longleftarrow z_i \tau_{ij} + z_j r_j$
4. $T_k \longleftarrow$ GetMWB(G, b, w)
5. $\mathbf{y} \longleftarrow \mathbf{z}/w(T_k)$
6. $\theta \longleftarrow \boldsymbol{\epsilon}^T \cdot \mathbf{y}$
7. **return** θ

In Algorithm 3, we provide the detailed description of the GetUB algorithm for computing an upper bound on the lifetime of the given MLDA instance and node energy prices $\boldsymbol{\phi} \in \mathbb{R}^{|V|}$. Recall that the shadow prices for the constraints of the linear program in (10) correspond to the negated energy prices of the nodes. Let $\mathbf{z} = \max\{-\boldsymbol{\phi}, \mathbf{0}\}$. Consider the edge weight function $w : E \longrightarrow \mathbb{R}$ that corresponds to $\mathbf{z}$, i.e. $w_{ij} = z_i \tau_{ij} + z_j r_j$ for each edge $ij \in E$. Let $T_k \in \mathcal{T}$ be a minimum weight branching for the edge weights $\mathbf{w}$. Observe that each row of $\mathbf{A}^T \cdot \mathbf{z}$ is equal to the energy cost of the corresponding branching, with energy prices given by $\mathbf{z}$, which in turn is equal to the weight of the same branchings under the edge weights $\mathbf{w}$. Therefore,

$$\mathbf{A}^T \cdot \mathbf{z} \geq w(T_k) \cdot \mathbf{1}. \tag{13}$$

Consequently, the vector $\mathbf{y} = \max\{-\boldsymbol{\phi}, \mathbf{0}\}/w(T_k)$ is a feasible solution to the dual linear program in (11). Moreover, $\boldsymbol{\epsilon}^T \cdot \mathbf{y}$ is an upper bound on the system lifetime for the given MLDA instance and energy prices $\boldsymbol{\phi}$.

The RSM–MLDA algorithm, computes at each iteration a basic feasible solution $\mathbf{x}$ for the primal linear program in (10), together with a vector of node energy prices $\boldsymbol{\phi}$. By calling the **GetUB** routine with parameters $\mathfrak{I}$ and $\boldsymbol{\phi}$, we obtain an upper bound on the optimal system lifetime. At each iteration of the RSM–MLDA algorithm, we use the smallest such upper bounds computed so far as our best upper bound on the system lifetime. Therefore, we can continuously assess the distance of the current feasible solution $\mathbf{x}$ from optimality, and we can terminate the RSM–MLDA algorithm early if that distance is below some desired threshold.

V. Experimental Results

We present experimental evaluation of the performance of our RSM–MLDA algorithm. We consider sensor networks in which sensors are uniformly distributed in a $50m \times 50m$ field while the base station is fixed at location $(45, 45)$. The network size n is set to take values 10–100 in multiples of 10. We generate 20 random networks for each network size. We use the first order radio energy model used in [8]. A sensor consumes $50nJ/bit$ to run the transmitter and receiver circuit, and $100pJ/bit/m^2$ for the transmitter amplifier. The packet size is 1000 bits. With these parameters, $r_i = 5mJ$ and $\tau_{ij} = 5 + 0.1 \cdot d_{i,j}^2 \;\; mJ$ for each sensor node i and j. Each sensor has $1J$ initial available energy. All of our experiments were done in Matlab running on a standard desktop PC.

First, we evaluate the practicability of our algorithm. We see in Fig. 1 that the RSM–MLDA algorithm takes a considerable number of iterations to reach the optimal continuous lifetime when the network size n becomes big, while in Fig. 2 we see that it gets within a small percentage of the optimal lifetime with few iterations for all the network sizes n considered in our experiments; e.g. for networks of size $n = 100$, it takes 2300 and 4200 iterations on average to reach 90% and 95% of the optimal lifetime, respectively. In practice, for medium-to-large WSNs, it may be acceptable to terminate the RSM–MLDA as soon as it reaches a user defined threshold for the approximation ratio, e.g. 95%. In such scenarios, we believe the RSM–MLDA would be very useful.

We discussed how to compute an upper bound of the system lifetime during the execution of the RSM–MLDA algorithm. Such an upper bound can be used to terminate the algorithm early, as soon as the estimated approximation ratio exceeds a user defined threshold. We evaluate the performance of the RSM–MLDA algorithm with early termination as well as the quality of the estimated upper bound on the lifetime, for two threshold values: 80% and 90%, see Figs. 3 and 4. We see that the approximation ratio achieved at early termination can be higher than the chosen threshold — as shown in Fig. 3, on average the approximation ratio achieved is 92% and 95% for threshold 80% and 90%, respectively. Fig. 4 shows the average number of iterations for the estimated approximation

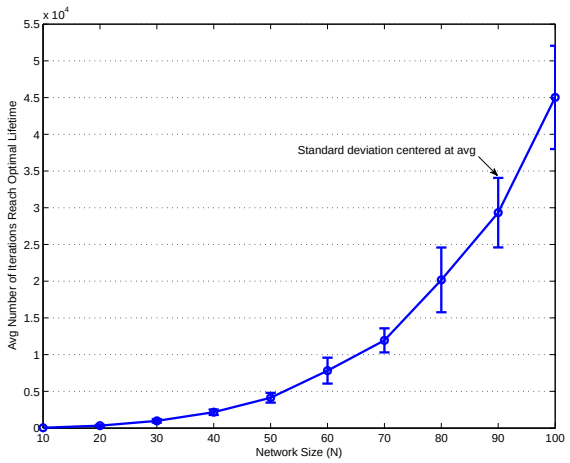

Fig. 1. Average number of iterations for the RSM–MLDA algorithm to reach the optimal continuous lifetime.

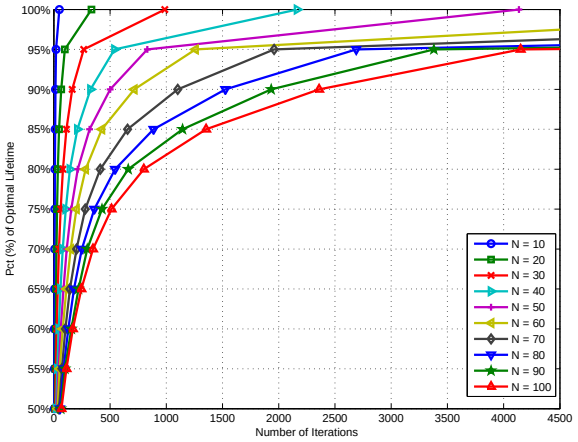

Fig. 2. Average approximation ratio vs number of iterations of the RSM–MLDA algorithm for different network sizes n.

ratio to exceed the chosen threshold — RSM–MLDA exceeds that threshold within a small number of iterations.

Second, we experimentally compare the RSM–MLDA algorithm with an iterative algorithm proposed by Stanford and Tongngam [13], which we call the GK algorithm, since it is based on the Garg–Konemann approach [6]. We run both algorithms on all the generated networks of various sizes n. For the GK algorithm, we use $\epsilon = 0.1$. We allow each algorithm to iterate at most $5n$ times.

Fig. 5 shows the approximation ratio achieved by the GK and RSM–MLDA algorithms – each point in the figure corresponds to a network instance. The approximation ratio is calculated as the lifetime achieved at termination over the optimal lifetime for that network. We see that RSM–MLDA outperforms GK in every network instance. Moreover, the minimum number of aggregation trees generated by GK at termination is $3, 69, 146, 200, 250, 300, 350, 400, 450, 500$ for all networks of size $n = 10, 20, \cdots, 90, 100$, respectively. Hence, the number of aggregation trees used by the GK

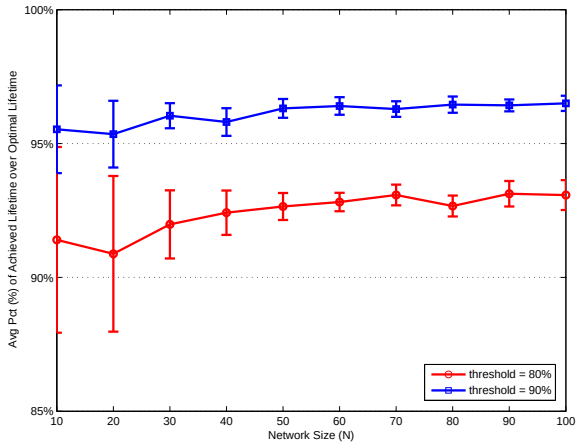

Fig. 3. Average approximation ratio achieved by the RSM–MLDA algorithm with early termination.

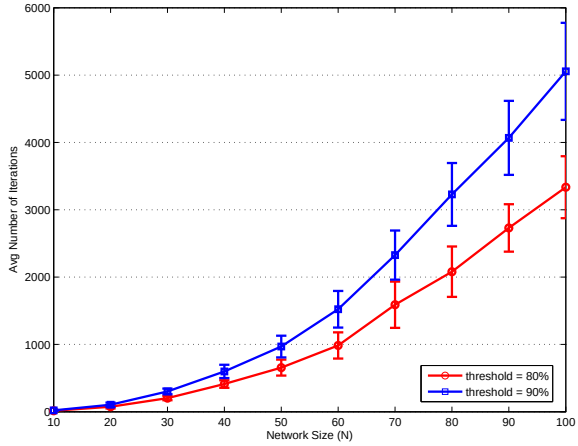

Fig. 4. Average number for iterations for the RSM–MLDA algorithm with early termination.

algorithm is close to the number of iterations it performs. Given the limited computational resources of the sensors, and the large number of iterations for GK to converge for small ϵ, distributing the relevant information about a large number of aggregation trees to the sensors is challenging and limits the usability of the GK algorithm. As we see in Fig. 5, GK achieved an average approximation ratio of 30% for networks of 50 nodes by 250 iterations – one needs to use 250 trees to get 30% of the optimal lifetime for the GK algorithm vs. up to 50 trees to get over 75% of the optimal lifetime for the RSM–MLDA algorithm.

VI. Conclusions

In this paper we considered the problem of Maximum Lifetime Data Gathering with in-network Aggregation (MLDA) in wireless sensor networks. We describe a combinatorial iterative algorithm for finding an optimal continuous solution that consists of up to $n-1$ aggregation trees and achieves lifetime T_o, which depends on the network topology and initial energy available at the sensors. We obtain an α–approximate

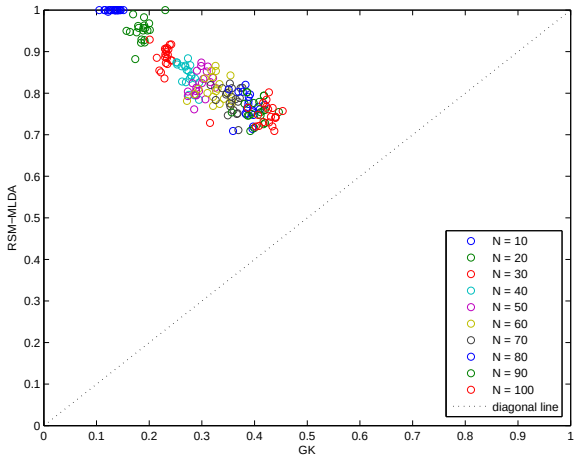

Fig. 5. Approximation ratio achieved by the GK and RSM–MLDA algorithms for each network instance.

optimal integral solution by simply rounding down the optimal continuous solution, where $\alpha = (T_o - n + 1)/T_o$. Since in practice $T_o \gg n$, $\alpha \approx 1$. We get asymptotically optimal integral solutions to the MLDA problem whenever the optimal continuous solution is $\omega(n)$. We also demonstrate the practicability of the proposed algorithm via extensive experimental results.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubermanian, and E. Cayirici. A survey of sensor networks. *IEEE Comm. Magazine*, 40:102–114, 2002.

[2] R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz. Local ratio: A unified framework for approximation algorithms. *ACM Computing Surveys*, 36(4):422–463, 2004.

[3] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[4] J. H. Chang and L. Tassiulas. Fast approximate algorithm for maximum lifetime routing in wireless ad-hoc networks. In *Networking 2000*, volume LNCS 1815, pages 702–713, 2000.

[5] K. Dasgupta, K. Kalpakis, and P. Namjoshi. Improving the lifetime of sensor networks via intelligent selection of data aggregation trees. In *Proc. of the CNDS'03*, Orlando, Florida, January 2003.

[6] N. Garg and J. Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[7] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Mobile Networking for Smart Dust. In *Proc. of 5th ACM/IEEE Mobicom Conference*, 1999.

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

[9] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Verlag, 1991.

[10] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan. Low-power wireless sensor networks. In *VLSI Design*, 2001.

[11] J. Rabaey, J. Ammer, J. da Silva Jr, and D. Patel. PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes. In *Proc. of the IEEE Computer Society Annual Workshop on VLSI*, 2000.

[12] N. Sadagopan and B. Krishnamachari. Maximizing data extraction in energy-limited sensor networks. In *Proc. of INFOCOM*, 2004.

[13] J. Stanford and S. Tongngam. Approximation algorithm for maximum lifetime in wireless sensor networks with data aggregation. In *Proc. of the SNPD'06*, pages 273–277, Washington, DC, USA, 2006.

[14] R. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[15] Y. Xue, Y. Cui, and K. Nahrstedt. Maximizing lifetime for data aggregation in wireless sensor networks. *Mobile Networks and Applications*, 10(6):853–864, 2005.