Collecting Data in Ad-Hoc Networks with Reduced Uncertainty

Liron Levin [a] Alon Efrat [b] Michael Segal [c]

[a] *Dept. of Communication Systems Engineering, Ben-Gurion University. Email:* levinlir@gmail.com

[b] *Dept. of Computer Science, The University of Arizona, USA. Email:* alon@cs.arizona.edu

[c] *Dept. of Communication Systems Engineering, Ben-Gurion University. Email:* segal@cse.bgu.ac.il

Abstract

We consider the data gathering problem in wireless ad-hoc networks where a data mule traverses a set of sensors, each with vital information on its surrounding, and collects their data. The mule goal is to collect as much information as possible thereby reducing the information uncertainty but in the same time avoid visiting some of the nodes to minimize its travel distance. We study the problem when the mule travels over a tree or a tour and propose a 3-approximation algorithm that minimizes both the information uncertainty and travel distance. We also show the applicability of our approach for solving data collection problems in varying domains such as temperature monitoring, surveillance systems and sensor placement. Simulation results show that the proposed solution converges to the optimal for varying set of topologies, such as grids, stars, linear and random networks.

Key words: Data gathering; mule traversal; optimization; approximation algorithm.

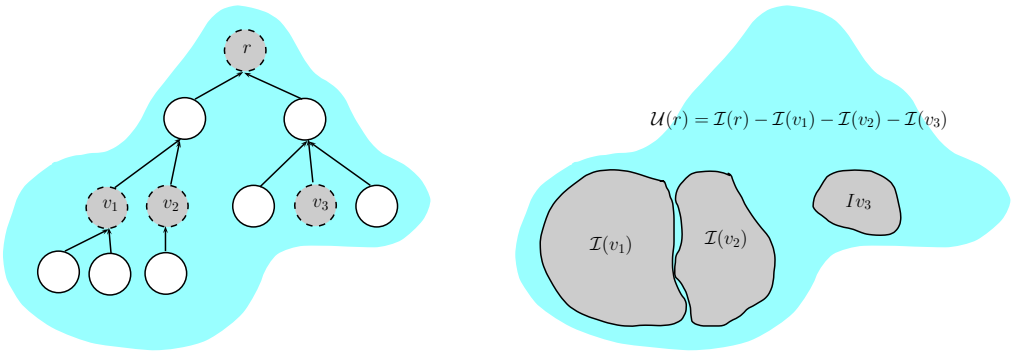
1 Introduction

Ad-hoc networks have become a popular research area both theoretically [32] and practically [15, 24, 28]. One of the most interesting and practical class of applications for sensors networks is agriculture and environmental monitoring [8, 22], where wireless nodes are scattered over a geographical area and form a dynamic, infrastructure-less ad-hoc network. Typical monitoring scenarios consist of two stages. In the first step, the sensors periodically sense their surroundings, collect data and process it if needed. In the second step,

the collected data is delivered to a base station using either *local* or *global* data collection process. In local data collection [21], the nodes use multi-hop communication over a static topology [5] or dynamic topology [17] to relay their message to the base station. This communication schema is most suitable when the distribution of nodes is dense, since the close proximity between nodes reduces the cost of transmitting and receiving messages [23]. In global data collection [1,6,21,25,26], the nodes are visited by a traveling *mule*, which uses short-range wireless communication to collect data from nearby nodes. The mule aggregates the information received from the nodes and delivers it to the base station. The data mule approach is most suitable when the network topology is sparse, the distance to the nearest node or base station is too large, or when the communication infrastructure between nodes is unstable. In addition, this approach reduces the responsibility for message routing from the nodes thereby minimizing their processing power. Most papers that use the data collector mule focus on energy efficiency [1, 16] or travel time optimization [30] without considering the quality of the collected data. Since data in each node is closely linked to the monitored area, deciding which data is significant and which is redundant can greatly contribute to the performance of the data collecting algorithm. For example, consider an environmental monitoring system with large distribution of sensors in a specific geo-location. The added knowledge from visiting one node in the area might contribute significantly to the overall understanding of the environment. However, the knowledge obtained by visiting additional nodes in nearby locations may not be worth the time and transmission cost of the visit. Thus, the value from visiting a node is not a constant and depends on the nodes that were visited before it. In this paper we study a dual-objective optimization problem, where the goal is to minimize the mule's traveling distance while minimizing the amount of information uncertainty caused by not visited a subset of nodes by the mule.

Our contribution. We present a framework for solving the mule problem when data each node sense is correlated with its neighbors. Our novel technique is used to find a tour or a tree under varying objective functions and has a 3-approximation ratio guarantee.

Outline of the paper. The rest of the paper is organized as follows: In Section 2, we present the system model, provide motivation and present the Mule Tree Problem (*MTP*) and Mule Cycle Problem (*MCP*). In Section 3 we discuss about problem background and previous work. We show a dual-primal formulation of the problem in Section 4, develop our 3-approximation algorithm for MTP and MCP in Section 5 and also show how to apply the algorithms for multiple optimization functions. We numerically evaluate the performance of our solutions against several common algorithms in Section 6 and conclude in Section 7.



(a) The communication tree. Visited nodes are marked by dotted circles

(b) Area covered by the visited nodes

Figure 1. Communication tree and covered areas.

2 Network Model

Let $G = (V, E, r)$ be a complete graph embedded in the Euclidean plane, where V is the set of wireless nodes ($|V| = n$) and E is the set of undirected edges between nodes. Here r is the base station and the other nodes represent sensors. For each $v \in V$, let $i(v) \in \mathbb{R}_+$ be the amount of data sensed by v and for each $e \in E$, let c_e be the mule's cost of traversing an edge e. Consider a mobile entity with wireless capabilities called mule that visits a subset of nodes $V_m \subseteq V$ and collect the information they sense. The mule tour or cycle starts at r and traverse along a subset E_m of of the edge of E. The mule can decide to skip over some nodes $\overline{V_m} = V \setminus V_m$ and to absorb a penalty $\varphi(\overline{V_m}) \in \mathbb{R}_+$. The value of the penalty reflects the data the mule would not collect. It is worth noting that the mule does not always have to visit a node to know its stored value; in many cases, this value could be inferred from knowing values in different part of the graph. For example, in aggregation binary tree, where nodes store the sum of values of their descendants, knowing the sum at a node v and one of its children would reveal the value of the other child (by merely performing subtraction).

A natural question arises whether the mule should visit all nodes, or skip over some and absorb the penalty. We ask how to find a tour that is not too long, but contains enough monitored data. Formally, the problem is defined as follows:

The Mule Tree/Cycle Problem (MTP/MCP)
Input: *Graph $G = (V, E)$, cost c_e per edge, information sensed by each node $i(v)$ and a root r*
Output: *A tree $T_{mule} = (V_m, E_m)$ or a cycle $C_{mule} = (V_m, E_m)$*
Objective: $\min(\varphi(\overline{V_m}) + \sum_{e \in E_m} c_e)$

In our problem φ represents the penalty function of not visiting the nodes of $\overline{V_m}$. We present a framework for approximating the mule problem for a variety of penalty functions. The approximation ratio holds as long as the following condition is fulfilled:

Condition 1. *For any two disjoint node sets, S_i and S_j, $\varphi(S_i) \leq \varphi(S_i \cup S_j)$.*

Specifically, we show that different penalty functions cover a variety of practical scenarios and present two concrete examples for the mule problem. First, we show an example related to environmental monitoring:

Temperature monitoring. For simplicity, we discuss the case where each sensor measures temperature (though, of course, this is not a restriction). Let $i(v)$ be the average temperature in the area covered by v. The objective is for the base station r to "know" the temperature measured by *each* sensor, but this goal might require too many relays (forwarding) in a multi-hoop fashion, which might be beyond accessible network resources such as battery life and channels capacities. Hence we opt to use an aggregation tree, and once needed (as formally articulated below), send the mule to retrieve to r more measurements from a (carefully-selected) subset of the sensors, so the uncertainty will decrease below a desired threshold.

Let $T = (V, E', r)$ be a communication tree rooted at r. Along its edges, sensors sends (summarizations of) measured temperature samples between adjacent nodes toward r. We assume channel capacity and sensors batteries' life are limited, hence transmitting all measurement using large messages in T is not prohibited. Instead, we assume that only summarizations are sent. Let $\mathcal{D}(v)$ be the set of the descendants of v in T. Due to the capacity limitation, node v can only send $i(v)$ and its descendant data $\sum_{u \in \mathcal{D}(v)} i(u)$ to its parent in T. We define this process as data aggregation. After data aggregation is completed by all nodes, each node v maintains the aggregated information, $\mathcal{I}(v) = \sum_{u \in \mathcal{D}(v)} i(u) + i(v)$, which represents what v knows about the temperature in its surrounding. When the mule visits a node v, it can learn the local information sensed $i(v)$ and the aggregated information $\mathcal{I}(v)$. From this information, the mule can infer on the data discrepancy between v's data to its descendants that was visited, $\mathcal{U}(v) = \mathcal{I}(v) - \sum_{u \in \mathcal{D}(v) \cap V_m} \mathcal{I}(u)$. Since $\mathcal{U}(v)$ decreases as the amount of data we collect increases, we define this measure as information uncertainty. $\mathcal{U}(v)$ is illustrated in Figure 1. In Figure 1a, we have the input tree T, visited nodes $V_m = \{r, v_1, v_2, v_3\}$, and $\mathcal{D}(r) = \{v_1, v_2, v_3\}$. The amount of information per node v, $\mathcal{I}(v)$, is depicted in Figure 1b. Intuitively, by visiting only 3 nodes, we learn the temperature of almost the entire area. As the number of visited nodes becomes higher, the amount of uncertainty decreases. Specifically, the following penalty functions can be used to

minimize the uncertainty:

$$\varphi_1(\overline{V_m}) = \frac{\sum_{v \in V_m} \mathcal{U}(v)^2}{|V|}$$

$$\varphi_2(\overline{V_m}) = \max_{v \in V_m} \mathcal{U}(v)$$

The first function guarantees that on average we learn sufficient information from each node (w.l.o.g the normalization factor is removed to simplify the calculation). The second function guarantees that sufficient information is known on every area in the graph. We provide an example for MCP with penalty function $\varphi_1(\overline{V_m})$. The input communication graph $G = (V, E)$ with 4 nodes, r, v_1, v_2 and v_3, is depicted in Figure 2. The tree T, on which the messages get aggregated, is depicted by the non dotted directed edges in the figure. The dotted edges do not belong to T but can be used by the mule to traverse the graph. Each edge is marked with the Euclidean cost of traversing it. Each node covers the Euclidean area $\mathcal{I}(v)$, which is is depicted by surrounding circles in the middle figure. The cost of selecting each subset of nodes S that includes the root r is depicted in table from the right. The optimal mule tour is traversing nodes r and v_2, with total cost of 88.

Surveillance system. We explore the following penalty function:

$$\varphi_3(\overline{V_m}) = \sum_{v \in \overline{V_m}} f(v),$$

where $f(v)$ is a monotonic increasing function, and $\overline{V_m} = V \setminus V_m$. Given n surveillance sensors partitioned to k different regions, we would like to find a mule tour that visits as many regions as possible. To solve the surveillance problem we use φ_3 and set $f(v)$ to $\alpha \in \mathbb{R}_+$ if all nodes from specific region belong to $\overline{V_m}$ or 0 otherwise. The penalty function ensures that sufficient area is under surveillance. Another option is to set $f(v) = |\overline{V_m}|i(v)$, and then we get, $\varphi_3(\overline{V_m}) = |\overline{V_m}| \sum_{v \in \overline{V_m}} i(v)$, which ensures that in addition to information loss per node we don't skip over too many nodes. This penalty function can also incorporate the spatial locality by correlating the definition of $i(v)$ with the number of the readings and frequency of users per region. For example, by setting $i(v) \propto P(k)$, where $P(k)$ is the probability that region k is visited by some smartphone users, we ensure that areas with high visit frequency will be covered by the mule [33].

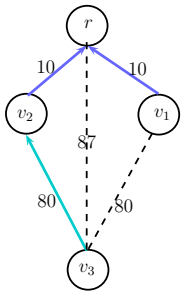
Sensor placement with reduced uncertainty Another practical scenario where the mule algorithm can be used is sensor placement with reduced uncertainty. Assume we have discrete set of rooms, where the temperature in the rooms varies according to some probabilistic distribution function P. Our goal is to distribute a set of nodes in different rooms, such that the temperature uncertainty and the travel distance between the nodes is minimized (i.e., we

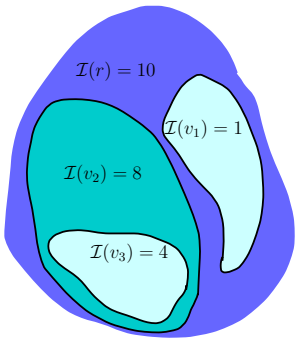
can choose to leave some rooms empty). Let $\overline{V_m}$ be the set of rooms which are not covered by the sensors, T be a selected placement of nodes, and $\mathcal{T}(v)$ be a random variable representing the temperature sensing of sensor v in a specific room. The conditional probability that the temperature sensing in room v is equal to t, $\Pr(v = t|T)$, is defined as the probability that v's temperature is equal to t given the total temperature measured so far $\sum_{v \in T} i(v)$. The uncertainty function of not visiting those rooms can be defined is follows:

$$\varphi_4(\overline{V_m}) = \sum_{v \in \overline{V_m}} H(\mathcal{T}(v)|T),$$

where $H(\mathcal{T}(v)|T)$ is the conditional entropy of $\mathcal{T}(v)$ given a specific placement T. Here we use entropy since it describes well the temperature uncertainty of the measured rooms (see [19, 31]).

Clearly the complexity of the solution depends on the selection of P, and the exact definition of the conditional entropy. For example, if P is taken from the discrete uniform distribution (e.g., temperature in each node varies between $0°$ to $10°$), then the conditional probability $\Pr(v = t|T)$ is equal to one divided by the number of restricted partitions of $\sum_{v \in T} i(v) - \sum_{v \in \overline{V_m}} i(v) - t$ with size of at most $|\overline{V_m}| - 1$, which can be polynomially calculated [2]. As long as the computation time of φ_4 is polynomially bounded, other more complex probability distribution functions can also be used. For example, Gaussian process or a function that depends on the temperature covariance matrix [20].





V_m	$\sum_{v \in V_m} \mathcal{U}(v)^2$	$\sum_{e \in E_m} c_e$	Total
$\{r\}$	$10^2 = 100$	0	100
$\{r, v_2\}$	$(10-8)^2 + 8^2 = 68$	20	88
$\{r, v_1\}$	$(10-1)^2 + 1^2 = 82$	20	102
$\{r, v_1, v_2\}$	$(10-8-1)^2 + 8^2 + 1^2 = 66$	30	96
$\{r, v_3\}$	$(10-4)^2 + 4^2 = 52$	174	226
$\{r, v_3\}$	$(10-4)^2 + 4^2 = 52$	174	226
$\{r, v_2, v_3\}$	$(10-8)^2 + (8-4)^2 + 4^2 = 36$	177	213
$\{r, v_1, v_3\}$	$(10-4-1)^2 + 4^2 + 1^2 = 42$	177	219
$\{r, v_1, v_2, v_3\}$	$(10-8-1)^2 + (8-4)^2 + 4^2 + 1^2 = 34$	180	214

Figure 2. Physical example for the Mule Problem. The left image depicts the input graph with traveling cost and information per node $i(v)$ and the area each node sense is illustrated in the middle image. The right table summarizes the mule cost of selecting each subset of nodes.

3 Previous Work

Most of the work on data mules in ad-hoc networks is focused on finding efficient algorithms by means of energy or scheduling. For example, in [6],

the authors propose a schema for collecting data from sensors that transmit messages at a specific rate. They define the time a collector spends in collecting messages as the stability of the system, provide sufficient conditions for network stability, and derive upper bounds on the message delay of specific algorithms. Another interesting model was proposed in [10], where a collector with a bounded travel path must visit a set of sensors. A collector successfully receives a message only when it is within the transmission range of a sensor. The optimization criteria is to minimize the sum of transmission cost of all nodes and their solution involves reducing the problem to Traveling Salesman Problem with Neighborhoods [3]. Another approach was suggested in [26], where the authors considered a three-tier architecture that consists of sensors, transport agents (mules) and a top tier wireless access network (WAN). Assuming the mules perform a random walk, they analyze the time to collect data from all sensors and reach the top tier. In a subsequent paper [16], the authors extend their analysis with additional performance and model metrics, such as data transfer, latency, power and mobility pattern. Other work includes an experiment of the mule's three-tier architecture using real Mica2 sensors running the TinyOS operating system [11].

The foundation of the algorithms proposed in this paper lies in a general framework for solving constraint degree forest problems, which is also used to solve the Prize Collecting Traveling Salesman Problem (PCTSP) [4]. In PCTSP, a salesman must travel to a known number of sites using the shortest tour, but it may skip over some sites and pay a constant penalty per site omitted. PCTSP was subject to several studies [12, 14] and has a 2-approximation algorithm. As opposed to the dynamic penalty of not visiting nodes in the mule problem, in PCTSP the penalty of not visiting a subset of the sites is equal to the sum of penalties of the sites. Similar to the approach we show in this paper, there were several attempts to solve the problem for more general penalty functions. For example, the authors in [27] show a 3-approximation algorithm to the problem when the penalty function is sub-modular (for any two set S and T $\varphi(S) + \varphi(T) \geq \varphi(S \cup T) + \varphi(S \cap T)$); this assumption is not valid for the penalty function we propose.

4 Problem Formulation

At the first step of the algorithm, we express MTP as an ILP by using a variable x_e for each edge, and an boolean variable, γ_C, for any set $C \subset V$. This variable specifies whether the nodes of C are all excluded or all included from the mules tree. (i.e., $\gamma_C = 1$ iff C is the visited set $\overline{V_m}$). The formulation

is as follows:

$$\begin{aligned}
\min \quad & \sum_{e \in E} x_e c_e + \sum_{C \subset V; r \notin C} \gamma_C \varphi(C) & \\
\text{subject to:} \quad & \sum_{e \in \delta(S)} x_e + \sum_{S \subseteq C} \gamma_C \geq 1 & S \subset V; r \notin S \\
& \sum_{C \subset V; r \notin C} \gamma_C \leq 1 & \\
(\text{MTP} - \text{IP}) \quad & x_e \in \{0,1\} & e \in E \\
& \gamma_C \in \{0,1\} & C \subset V; r \notin C
\end{aligned}$$

The formulation is standard for constraint tree problems. Let $\delta(S)$ be the set of outgoing edges from S. The linear constraint enforces that for each set of nodes S, one of the two conditions holds: S has at least one outgoing edge or S is fully contained in the excluded set $\overline{V_m}$, i.e., it must be part of the tree or excluded. The target function is to minimize the sum of edges in the tree and the cost of the excluded set. To effectively solve the problem we create a linear programming relaxation MTP-LP by replacing the integer constraints with $x_e \geq 0$ and $\gamma_C \geq 0$.

The dual of the LP, which uses variables y_S to represent the set S and λ for the set $\overline{V_m}$, is defined as follows:

$$\begin{aligned}
\max \quad & \sum_{S \subset V; r \notin S} y_s - \lambda & \\
\text{subject to:} \quad & \sum_{S | e \in \delta(S)} y_S \leq c_e & e \in E \\
(\text{MTP} - \text{DP}) \quad & \sum_{S \subseteq C} y_S - \lambda \leq \varphi(C) & C \subset V; r \notin C \\
& y_s \geq 0 & S \subset V; r \notin S \\
& \lambda \geq 0 &
\end{aligned}$$

In the dual formulation, the variable y_S represents the cost of node set S. Let e be an edge that has one endpoint in S. The first constraint ensures that the total cost of this set is at most c_e and the second constraint ensures that the

sum of dual variables for any set C is at most $\varphi(C)$.

5 Approximation algorithms for the Mule problem

In this section, we study two algorithm for the mule problem. In Subsection 5.1, we present a 3-approximation when the mule travels through a tree. We extend the algorithm to produce a tour in Subsection 5.2. We conclude this section with a discussion on practical implementation of the global algorithm in a distributed sensor environment.

5.1 Algorithm for Mule Tree Problem

In this section we present Algorithm 1, a 3-approximation dual-primal algorithm for MTP. We begin with some basic notations. We say that an edge $e \in \delta(S)$ is *tight* if the sum of the dual variables of the sets for which e is an outgoing edge equal to c_e, i.e., $\sum_{S|e\in\delta(S)} y_S = c_e$. In each step of the algorithm, every set of nodes can be in one of the following states: *rooted*, a set that contains the root, *dead*, a set that is not included in the tree, and *active*, a set that might be included in the final tree. The dead sets, $C_1, C_2, ..., C_k$ are ordered by the round they were declared dead. Algorithm 1 modify the tree constraint dual-primal approximation algorithm from [12] with a new condition in Line 17 and a tree finalization technique in Line 23. We show that this modification only changes the approximation ratio from 2 to 3. First we prove the dual-primal feasibility of the result of the algorithm:

Lemma 2. *After Algorithm 1 completes, the primal problem is feasible*

Proof. The output of the algorithm is a rooted tree and a set of dead sets, which represents $\overline{V_m}$. Thus, all primal constrains are satisfied and the solution is feasible. □

Lemma 3. *After Algorithm 1 completes, the dual problem is feasible*

Proof. Clearly, at the beginning of the algorithm all dual constraints are satisfied since for each $S \in V$, y_S is set to be zero. In Algorithm 1, we have two cases: adding a new edge and declaring a set dead. Consider the former case, adding a new edge means that the dual constraint became tight and therefore the dual is still feasible. Now consider the latter case. If the cost of all dead sets does not exceed their penalty, we are done; otherwise, in a previous step two active sets S_i and S_j were merged, but their cost together with the cost of all sets that were declared dead has surpassed the total penalty for the

Algorithm 1: Mule Tree Problem Algorithm

Input: Graph $G = (V, E)$, root node r, and a penalty function φ.
Output: A tree $T = (V_m, E_m)$.

Start with a dual feasible solution by creating a corresponding set $S = \{v\}, \forall v \in V$, with $y_S = 0$, and set $\lambda = 0$.
Activate all sets except from the one that contains the root.
while *Some sets are active* **do**
    Simultaneously increase the weight y_S by equal amount for all active sets until one of the following happens:
    case *An edge e between an active set S_i to a set that is connected to r becomes tight*
        S_i becomes deactivated.
        S_i is merged with the set containing r.
        $E_m = E_m \bigcup \{e\}$.
    endsw
    case *An edge between two sets S_i and S_j becomes tight*
        The sets are merged to a new active set S_w, with $y_{S_w} = y_{S_j} + y_{S_i}$.
        $E_m = E_m \bigcup \{e\}$.
    endsw
    case *The cost of an active set S_i is equal to the penalty of of excluding this set i.e., $y_{S_i} \geq \varphi(S_i)$.*
        S_i is declared dead.
    endsw
    case *The cost of an active set S_i, including the cost of all other dead sets $\sum_{i=1}^{k} y_{C_i}$ is equal to the penalty of the union of those sets, i.e., $y_{S_i} + \sum_{i=1}^{k} y_{C_i} \geq \varphi(S_i \bigcup_{i=1}^{k} C_i)$.*
        S_i is declared dead.
        if $y_{S_i} + \sum_{i=1}^{k} y_{C_i} > \varphi(S_i \bigcup_{i=1}^{k} C_i)$ **then**
            $\lambda = \lambda - (y_{S_i} + \sum_{i=1}^{k} y_{C_i} - \varphi(S_i \bigcup_{i=1}^{k} C_i))$
        end
    endsw
    Let $C = \bigcup_{i=1}^{k} C_i$ and $y_C = \sum_{i=1}^{k} y_{C_i}$
    while $y_C < \varphi(C) - \lambda$ **do**
        Increase y_C until one of the following happens:
        case *An edge e between the rooted set and C becomes tight*
            $E_m = E_m \bigcup \{e\}$.
            Let C_j be the dead set that has e as an outgoing edge.
            Add all edges in C_j to E_m.
            Remove C_j from C.
        endsw
        case $y_C = \varphi(C)$
            yield break.
        endsw
    end
    Remove as many edges as possible from E_m while keeping the following conditions: a) All nodes that do not belong to dead components are connected to r, and b) If a node that belongs to a dead component C_i is mandatory to maintain condition a), then all other nodes in C_i must also be included in V_m. V_m is constructed from the endpoints of E_m.
end

combined set. Line 19 fixes this condition by increasing λ and keeps the dual solution feasible. □

Lemma 4. *After Algorithm 1 completes,* $\varphi(\overline{V_m}) \leq \sum_{S \subset V; r \notin S} y_S - \lambda$

Proof. A set D is declared dead, either if its cost is equal to its penalty or if summing its cost and the cost of all sets declared dead beforehand $C_1, C_2, ...C_j$ exceeded its penalty. Let C be the union of all dead set. For some penalty function φ, when adding set D to set C the total penalty is much higher than the current sum of dual variables, i.e., $\sum_{S \subseteq D \cup C} y_S < \varphi(D \cup C)$. This is resolved in line 23, where we increase the cost of the set C until either the cost of the dead set is equal to its penalty, or until an outgoing edge from the dead set becomes tight. In the former case we are done, since the cost of the dead sets is equal to their penalty. In the latter case, we add the node from the dead set and reapply the role. For example, see Figure 3, where the root r is connected to 2 nodes v_1 and v_2. The penalty of excluding either v_1 or v_2 is 0 but the cost of excluding both is 100. Since $\varphi_3(v_2) = \varphi_3(v_2) = 0$, after the first two steps of the algorithm both v_1 and v_2 are declared dead. Note that the if the algorithm terminates here, the approximation ratio is unbounded, since the primal cost is 100 while the cost of the dual $\sum_{S \subset V; r \notin S} y_S - \lambda$ is zero. This is fixed in line 23, where the dual variable of set $\{v_1, v_2\}$ increases until the edge between r and v_1 becomes tight. After this step, v_1 is added to the tree, and the algorithm terminates with the optimal solution (the tree is the edge between r to v_1). □

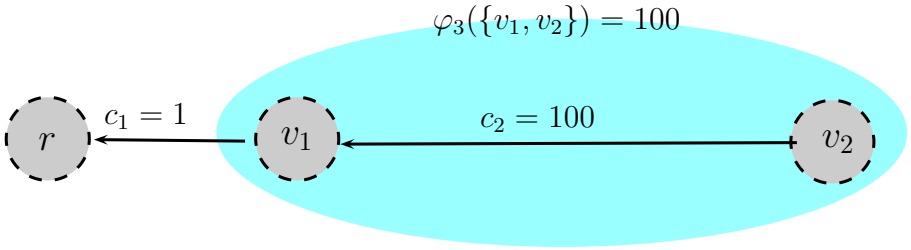


Figure 3. Example where additional steps must be taken to obtain better approximation of the optimal solution. The penalty of not selecting both v_1 and v_2 is 100 while the penalty of not selecting only v_1 or v_2 is zero.

Since we use consider general penalty functions, when merging two sets by a tight edge, the dual constraint $\sum_{S \subseteq C} y_S - \lambda \leq \varphi(C)$ might be violated. The following lemma shows that for penalty functions that obey Condition 1, the violation is at most by 2, so λ can be adjusted by a bounded value in Line 19 and this keeps the dual problem feasible.

Lemma 5. *There is always a value of* λ *that keeps the dual penalty constraint feasible.*

Proof. Let S_i and S_j be two components that were joined by a tight edge, and $C_1, C_2, ..., C_k$ be the dead components so far. By the construction of the

algorithm, we get that $y_{S_i} + \sum_{l=1}^{k} y_{C_l} < \varphi(S_i \bigcup_{l=1}^{k} C_l) - \lambda$ and $y_{S_j} + \sum_{l=1}^{k} y_{C_l} < \varphi(S_j \bigcup_{l=1}^{k} C_l) - \lambda$. When combining the equations we get:

$$y_{S_i} + y_{S_j} + 2\sum_{l=1}^{j} y_{C_l} < \varphi(S_i \bigcup_{l=1}^{k} C_l) + \varphi(S_j \bigcup_{l=1}^{k} C_l) - 2\lambda$$

We also have that: $\sum_{l=1}^{j} y_{C_l} = \varphi(\bigcup_{l=1}^{j} C_l) - \lambda$. Hence, we have:

$$y_{S_i} + y_{S_j} + \sum_{l=1}^{j} y_{C_l} + \varphi(\bigcup_{l=1}^{j} C_l) - \lambda < \varphi(S_i \bigcup_{l=1}^{k} C_l) + \varphi(S_j \bigcup_{l=1}^{k} C_l) - 2\lambda.$$

According to the definition of the penalty function φ, for any disjoint sets S_i and S_j, $\varphi(S_i) \leq \varphi(S_i \bigcup S_j)$. Therefore:

$$y_{S_i} + y_{S_j} + \sum_{l=1}^{j} y_{C_l} \leq$$

$$\varphi(S_i \bigcup_{l=1}^{k} C_l) + \varphi(S_j \bigcup_{l=1}^{k} C_l) - \varphi(\bigcup_{l=1}^{j} C_l) - \lambda \leq$$

$$2\varphi(S_i \bigcup_{l=1}^{k} C_l).$$

So the increment of λ is bounded by $2\varphi(S_i \bigcup_{l=1}^{k} C_l) - \varphi(\bigcup_{l=1}^{j} C_l) - \lambda$ and the proof is complete. □

Let $T^*_{MTP-LP}, T^*_{MTP-DP}$ be the optimal solutions to the primal and dual problems, T^* be the optimal solution to MTP, and y_S be the value of the dual variable after Algorithm 1 completes. We claim the following:
Lemma 6. *After Algorithm 1 completes,* $\varphi(\overline{V_m}) \leq T^*$.

Proof. Let C_k be the last set that was declared dead, by the construction of the algorithm we have:

$$\varphi(\overline{V_m}) = \varphi(\bigcup_{j=1}^{k} C_k) \leq \sum_{j=1}^{k} y_{C_j} - \lambda \leq \sum_{S \subset V; r \notin S} y_S - \lambda.$$

The first inequality follows from Lemma 4. Finally, using the primal-dual principle we get:

$$\sum_{S \subset V; r \notin S} y_S - \lambda \leq T^*_{MTP-DP} \leq T^*_{MTP-LP} \leq T^*.$$

□

We use the following theorem in our analysis:
Theorem 7. *[12, Theorem 2.4] After Algorithm 1 completes,* $\sum_{e \in E_m} c_e \leq 2T^*$.

We conclude with the following theorem:
Theorem 8. *Algorithm 1 is a* 3*-approximation to MTP problem.*

Proof. Combining Theorem 7 and Lemma 6 we get:

$$\sum_{e \in E_m} c_e + \varphi(\overline{V_m}) \leq 3T^*.$$

□

Following similar argument as in [12], the running time of Algorithm 1 is $O(n^2 \log n)$.

5.2 Algorithm for Mule Cycle Problem

In this section we show how to extend the ideas we developed in the previous subsection and show a 3 approximation algorithm when the target topology is a tour. Note that any α-approximation algorithm for MTP can be converted to a 1.5α-approximation algorithm by transforming the tree to a tour Christofides' 1.5 approximation algorithm for the traveling salesman problem [9].

First we formulate MCP as an ILP:

$$\begin{aligned}
\min \quad & \sum_{e \in E} x_e c_e + \sum_{C \subset V; r \notin C} \gamma_C \varphi(C) \\
\text{subject to:} \quad & \sum_{e \in \delta(S)} x_e + 2 \sum_{S \subset C; r \notin S} \gamma_C \geq 2 \; S \subset V; r \notin S \\
& \sum_{C \in V; r \notin C} \gamma_C \leq 1 \\
(\text{MCP} - \text{IP}) \quad & x_e \in \{0,1\} \qquad\qquad e \in E \\
& \gamma_C \in \{0,1\} \qquad\qquad C \subset V; r \notin C
\end{aligned}$$

γ_C is equal to 1 for the set of nodes that are not selected in the final tour and 0 otherwise. The second constraint enforces that each node included in

the tour gets visited at least twice (i.e., the solution is a tour). Similar to MTP-LP, the linear relaxation MCP-LP is formed by relaxing the second and third constraints as in MTP-LP. The dual problem is as follows:

$$\begin{aligned}
\max \quad & 2\sum_{S;r\notin S} y_S - \lambda \\
\text{subject to:} \quad & \sum_{S|e\in\delta_S} y_S \le c_e && e \in E \\
(\text{MCP}-\text{DP}) \quad & 2\sum_{S\subseteq C} y_S - \lambda \le \varphi(C) && C \subset V; r \notin C \\
& y_s \ge 0 && S \subset V; r \notin S \\
& \lambda \ge 0
\end{aligned}$$

We use the following algorithm to solve MCP:

Algorithm 2: Mule Cycle Problem Algorithm

Input: graph $G = (V, E)$
Output: A tour $T = (V_m, E_m)$, and a set of nodes $\overline{V_m}$ not in the tour.

1 Produce tree T' by running algorithm 1 on G with penalty $\frac{\varphi(S)}{2}$.
2 Produce a tour T by running a depth first traversal on T'.

We claim the following.

Lemma 9. *Algorithm 2 is a* 3*-approximation for MCP*

Proof. First note that the output of Algorithm 2 is a tour on the subset of V thus a valid solution to the primal problem. The dual solution also holds since we have $\sum_S y_S \le \frac{\varphi}{2}$. Let y_S^* be the value of set S in the optimal solution, y_S be the value of set S after running Algorithm 2, and T^* be the optimal solution to MCP-IP. The cost of the depth-first traversal is $2\sum_{e\in E_m} c_e$. Thus, we get:

$$2\sum_{e\in E_m} c_e + \varphi(\overline{V_m}) \le 4\sum_S y_S + 2\sum_S y_S \le 3(2y_S^*) \le 3T^*.$$

□

5.3 Penalty functions

We turn to prove that φ_1 and φ_2 are valid penalty functions and therefore can be used in Algorithms 1 and 2. The proof for φ_3 follows directly from its

definition.

Let C be the set of dead nodes in some iteration of the algorithm. We show that by increasing C by any number of nodes, the cost functions φ_1 and φ_2 are always not decreasing and therefore satisfy Condition 1. W.l.o.g we prove the claim on adding a single node v to C.

Lemma 10. *Cost function φ_1 satisfies Condition 1*

Proof. $\varphi_1(C)$ is equal to:

$$\underbrace{(\mathcal{I}(u) - \sum_{z \in \mathcal{D}(u) \setminus \{v\}} \mathcal{I}(z) - \mathcal{I}(v))^2 +}_{\#1}$$

$$\underbrace{(\mathcal{I}(v) - \sum_{l \in \mathcal{D}(v)} \mathcal{I}(l))^2}_{\#2} + \sum_{w \in V \setminus c \cup \{v,u\}} \mathcal{U}(w)^2.$$

Let u be the first ancestor of v that is does not belong to C. After removing v, the penalty $\varphi_1(C \cup \{v\})$ is:

$$(\mathcal{I}(u) - \sum_{z \in \mathcal{D}(u)} \mathcal{I}(z))^2 + \sum_{w \in V \setminus C \cup \{v,u\}} \mathcal{U}(w)^2.$$

Clearly the increase in term #1 is at least $\mathcal{I}(v)^2$, while the decrease in term #2 is at most $\mathcal{I}(v)^2$. Therefore, the target function is not decreasing. □

Lemma 11. *Cost function φ_2 satisfies Condition 1*

Proof. The penalty $\varphi_2(S_i)$ is equal to:

$$\max\{\underbrace{\mathcal{I}(u) - \sum_{z \in (\mathcal{D}(w) \setminus \{v\})} \mathcal{I}(z) - \mathcal{I}(v)}_{\#1}, \underbrace{\mathcal{I}(v) - \sum_{l \in \mathcal{D}(v)} \mathcal{I}(l)}_{\#2},$$

$$\underbrace{\max_{w \in (V \setminus C \cup \{v,u\})} \mathcal{U}(w)}_{\#3}\}.$$

After v is removed we get that the penalty $\varphi_2(C \cup \{v\})$ is:

$$\max\{\mathcal{I}(u) - \sum_{z \in \mathcal{D}(w)} \mathcal{I}(z), \max_{w \in (V \setminus C \cup \{v,u\})} \mathcal{U}(w)\}.$$

Since the increase in term #1 $\mathcal{I}(v)$, is larger than term that was removed #2, the target function increases. □

5.4 Practical implementation in a large distributed sensor network

The algorithms described in previous sections require solving a global optimization problem which is very hard to decentralize. Thus, it might not be practical in real sensor networks. We can adjust the algorithms to real networks by using the mule for calculation and adding the time dimension, i.e., the mule first traverse all nodes in the network and retrieve their data, then run the algorithm locally and decide which nodes are vital and which can be skipped. The mule can repeatably perform the above routine every predefined number of tours or when the variance in the information learned significantly changes. Further optimization of the algorithm can recalculate the information gain only for a specific areas or a subset of nodes and take the old measurement from the rest of the graph.

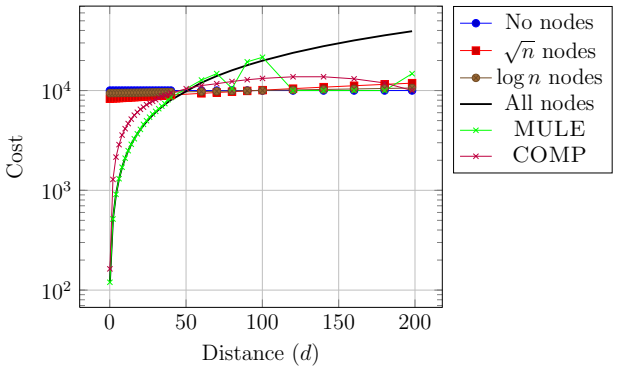
6 Simulations

In this section, we present numerical results corresponding to the analysis from previous sections. Using a custom simulator, we compare the performance of Algorithms 1 and 2, referred as MULE, to competitive algorithms on varying topologies, such as linear networks, stars, grids and random graphs. We choose a versatile set of rival algorithms. These include: distance greedy algorithm, a competitive incentive based algorithm COMP and when computationally feasible, the optimal algorithm OPT. The greedy algorithm selects $k \in [1, n]$ nodes and traverse them using the **optimal** tree or tour. To gain insights on the effect of node selection on the optimization function we set k to n, $\log n$, $\sqrt{n}$ and 1 (i.e., only the root is selected). The second competitive algorithm COMP, deliberately tries to optimize the optimization criteria by selecting edges that maximizes the information gain but minimizes the traveling distance. In each round, the algorithm expands the constructed tree or tour by finding the edge that minimize the travel cost but maximize the information gain. If the cost of this edge is positive (i.e., the travel cost is larger than the information gain), the algorithm randomly selects it with uniform distribution. The randomness is required since a distant node, which will never be visited by a greedy solution, may contains large amount of information. Intuitively, the mule must perform a leap of faith, and traverse an edge despite the decrease in the optimization function. The algorithm terminates when a round completes without selecting an edge. This algorithm was used since for the best of our knowledge, no competitive algorithms exist that optimize both information gain and travel distance. In practice, the algorithm yields good results under all topologies, but not as good as MULE and OPT.

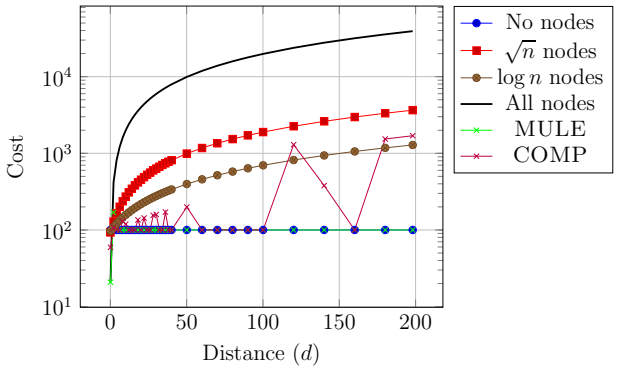
In our simulations, we assume sensors are stationary, and have limited wireless

Figure 4. Simulation results for temperature monitoring problem under different topologies. The 3-approximation algorithm is compared to several greedy functions that selects the closet nodes

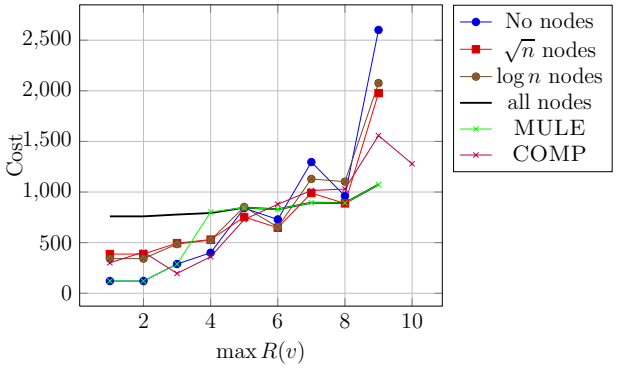
(a) Linear network - φ_1



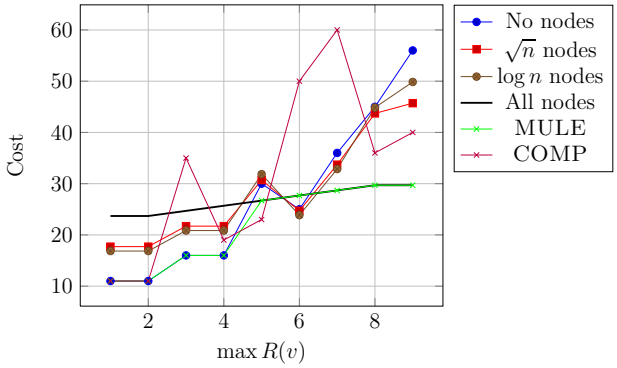
(b) Linear network - φ_2



(c) Star network - φ_1

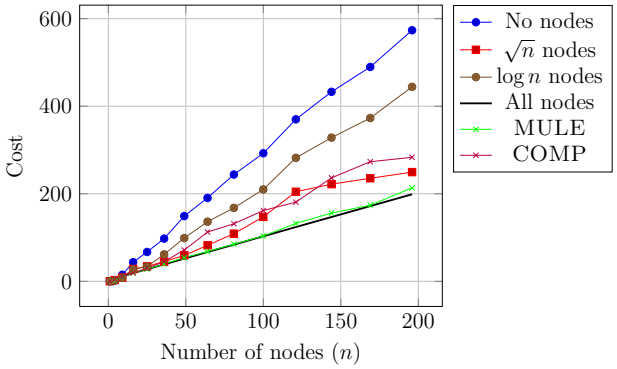


(d) Star network - φ_2

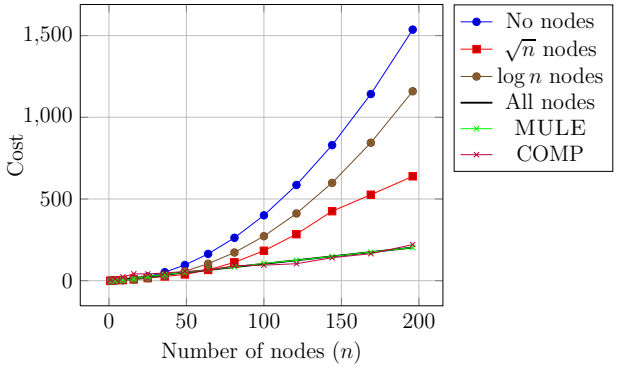


;

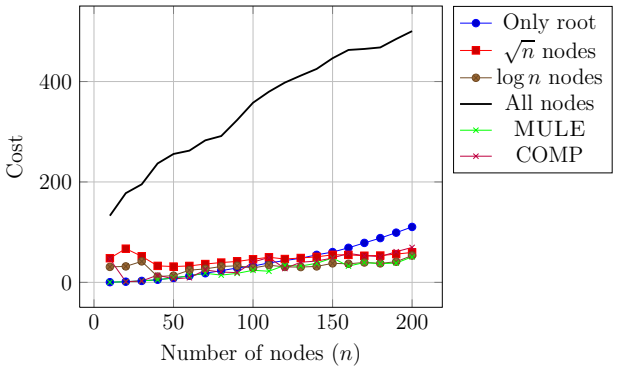
(e) Grid network - φ_1



(f) Grid network - φ_2



(g) Random network - φ_1



(h) Random network - φ_2

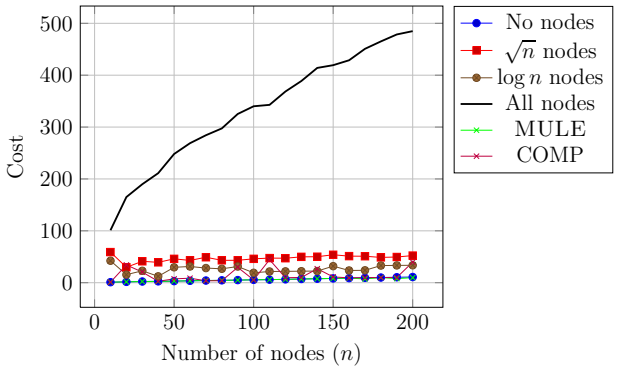
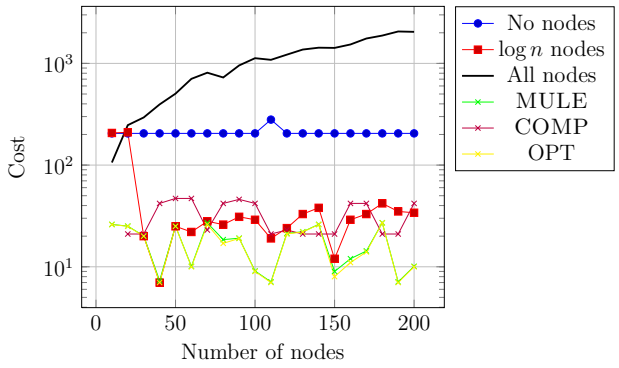
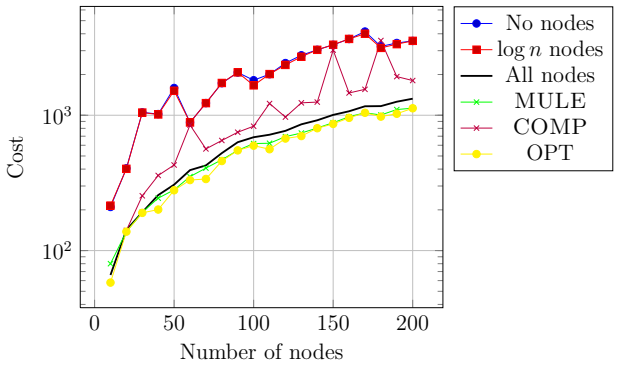


Figure 5. Simulation results on line topology using penalty function φ_3. The left figure shows the results on a topology where the mule has high incentive to skip visiting nodes farther from the root. The right figure shows the results where the farther the nodes, the higher the penalty of skipping it. For both figures, the Mule algorithm plot it closest to OPT plot.

(a) Linear network - φ_3



(b) Linear network - φ_3



transmission power [29, 30]. Under such settings, to receive the data stored in a sensor, the mule must visit the sensor's exact location in the network (i.e., the mule travels along the trajectory found by Algorithms 1 and 2).

Our first experiment studies the quality of our algorithms on linear ad hoc networks, which can be used for multiple data collection activities such as railway or structural monitoring [18]. To construct the topology, we distributed 100 nodes, each with $i(v) = 1$, on a straight line with equal distances d between the nodes. We calibrated d between 1 to 200, and compared between Algorithm 2 to the competitive algorithms described earlier. The results for penalty function φ_1 are shown in Figure 4a. When d is between 1 to 50, the optimal solution is to traverse all nodes using the optimal path (since the cost of edges is relatively negligible to paying the penalty of a node). Note that for lower values of d the ratio between the cheapest solution to the most expensive one is above 100, and for those, Algorithm 2 cost is exactly the same as the optimal solution. When d increases, the cost of selecting edges becomes significantly expensive to pay the penalty of all nodes, and thus the optimal solution is to select only the root. In this interval, Algorithm 2 cost converges to the optimal value. The zig-zag in the mule cost is explained by the fact that the difference between the no nodes cost and the all nodes cost is negligible, so the solution may change due to nodes distribution. When this difference becomes substantial the mule cost converges to the no nodes solution.

The results for penalty φ_2 are shown in Figure 4b. The results show similar trends as the first experiment, but since for a subset of nodes S, $\varphi_2(S)$ has lower penalty than φ_1, the shift between optimal solutions happens for lower values of d. For example, the cost of empty solution for φ_1 is 10^4 while the cost

for φ_2 is 10^2. As the value of d grows, the optimal solution changes between selecting all nodes to selecting only the root. For all values of d, Algorithm 2 cost converges to the optimal value as well.

In our second experiment, we examined the quality of Algorithm 2 on the star network. To construct the star, we placed the root r in the center on the graph, and equally distributed 10 nodes, each with constant distance from r, on the circumference of the circle covering r. The covered area of each node was randomly and equally distributed between 1 to some upper bound $\max \mathcal{I}(v)$. We compared the performance of Algorithm 2 when using φ_1 and φ_2 to the four rival greedy algorithms.

The results of running the algorithms for varying upper bounds when using φ_1 and φ_2 are plotted in Figure 4c and 4d, respectively. In those experiments, the value of the star radius was set to 99, which is high enough to drop nodes if their penalty is too low. Clearly, the cost of not selecting a node increases proportionally to its covered area. For small values of $\max \mathcal{I}(v)$, the solution that contains only the root has the minimum cost, while as $\max \mathcal{I}(v)$ increases, the optimal solution is to include all the nodes. The cost of Algorithm 2 always converges to the optimal value and never exceeds the optimal by more than 2 (which fits the theoretical bound of 3).

In the next experiment, we randomly and uniformly distributed a varying number of nodes on a 10×10 Euclidean grid. To construct the data gathering tree T, we start with a tree that contains only r and until all nodes belong to T, select a leaf v and a random integer $k \in [1, 5]$ and connect the closest k nodes that are not in T to v. The covered area per node is proportional to the density of nodes. The results for the cost functions φ_1 and φ_2 are plotted in Figure 4g and Figure 4h, respectively. For φ_1, Algorithm 1 converges to the optimal solution, achieving the best results in more than 60 percent of the samples, and for φ_2 Algorithm 1 has the same results as optimal solution for all samples.

The next experiment involves placing varying number of nodes in a random network with dimension $\sqrt{n} \times \sqrt{n}$. The grid topology can be used for reliable communication in ad hoc networks [7]. The covered area per node is proportional to the density of nodes in the graph (i.e., $\propto \frac{area}{n}$). The results when using φ_1 and φ_2 are plotted in Figures 4e and 4f, respectively. They show similar trends as in the previous experiments.

In our final experiment, we compared the result of the mule algorithm using penalty function φ_3 on the line topologies. In this topology, we were able to compute the optimal solution, which selects the closest k nodes to root such that the total penalty and the distance to the farthest node is minimized. Similar to previous experiment, we also compared the results to the naive al-

gorithms that select the closest k nodes to the root for varying values of k. To show that the mule algorithm behaves as expected under different circumstances we simulated two scenarios. In the first scenario, plotted in Figure 5a, we merge the node farthest from the root into a large clusters with big penalty. As intuition states, our results show that the mule algorithm, similar to the optimal algorithm, takes only one node from the large cluster and skips the rest of the nodes in it. Clearly, as the chart shows, the naive algorithms behaves very badly under those conditions since they collect unnecessary nodes. In the second scenario, plotted in Figure 5b, we randomly assigned nodes into groups of clusters, with penalty that depends on the number of nodes in each cluster. In this experiment, the mule algorithm have to work much harder to find the exact subset of nodes that yield optimal results. Opposite to the previous experiment, here selecting all nodes do yield good results, but the mule algorithm is still better by at least 10% than all other non-optimal algorithms.

7 Concluding remarks

In this paper, we developed a framework for finding near-optimal route for data gathering mule when the information gain of a node depends on the data collected so far. We identified a relationship between the mule problem to an extended variation of the Prize Collecting Traveling Salesman Problem, and presented a 3-approximation algorithm for it. This work is a first attempt towards a framework for solving data gathering problems that focus on the quality of data collected in addition to scheduling and power efficiency. Future extensions of our work could investigate how to solve the mule problem when the traveling distance is bounded (similar to the constrained orienteering problem [13]) or explore the case when multiple mules work together to collect data.

References

[1] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. In *IEEE Symposium on Computers and Communications*, pages 1096–1102, 2008.

[2] G. Andrews. *The Theory of Partitions*. Cambridge mathematical library. Cambridge University Press, 1998.

[3] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55:197–218, 1995.

[4] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[5] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, pages 1747–1757, 2005.

[6] G. D. Çelik and E. Modiano. Dynamic vehicle routing for data gathering in wireless networks. In *CDC*, pages 2372–2377, 2010.

[7] K.-H. Chen, C.-R. Dow, S.-C. Chen, Y.-S. Lee, and S.-F. Hwang. Harpiagrid: A geography-aware grid-based routing protocol for vehicular ad hoc networks. *J. Inf. Sci. Eng.*, 26(3):817–832, 2010.

[8] L. G. Chen and B. Xu. Towards efficient temperature monitoring and controlling in large grain depot. In *3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 881–885, 2006.

[9] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[10] D. Ciullo, G. D. Celik, and E. Modiano. Minimizing transmission energy in sensor networks via trajectory control. In *WiOpt*, pages 132–141, 2010.

[11] M. Conti, E. Gregori, and C. Spagoni. Motes sensor networks in dynamic scenarios. *International Journal of Ubiquitous Computing and Intelligence*, 1, 2006.

[12] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[13] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics (NRL)*, 34(3):307–318, 1987.

[14] S. Gutner. Elementary approximation algorithms for prize collecting steiner tree problems. In *Proceedings of the 2nd international conference on Combinatorial Optimization and Applications*, pages 246–254, 2008.

[15] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. Sensorscope: Application-specific sensor network for environmental monitoring. *ACM Transactions on Sensor Networks*, 6(2):1–32, 2010.

[16] S. Jain, R. Shah, W. Brunette, G. Borriello, and S. Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *MONET*, 11:327–339, 2006.

[17] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, 353:153–181, 1996.

[18] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN*, pages 254–263. ACM Press, 2007.

[19] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Robust sensor placements at informative and communication-efficient locations. *ACM Transactions on Sensor Networks*, 7(4):31:1–31:33, February 2011.

[20] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, June 2008.

[21] L. Levin, M. Segal, and H. Shpungin. Optimizing performance of ad-hoc networks under energy and scheduling constraints. In *WiOpt*, pages 11–20, 2010.

[22] Y. Ling and B. Xu. Minimizing energy with a risk based temperature monitoring protocol for wireless digital sensor network. In *Proceedings of the 2006 IET International Conference on Wireless, Mobile and Multimedia Networks*, pages 1–5, 2006.

[23] B. Liu, D. Towsley, and A. Swami. Data gathering capacity of large scale multihop wireless networks. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 124–132, 2008.

[24] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[25] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling a three-tier architecture for sparse sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 30–41, 2003.

[26] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: modeling and analysis of a three-tier architecture for sparse sensor networks. *Ad Hoc Networks*, 1(2-3):215–233, 2003.

[27] Y. Sharma, C. Swamy, and D. P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *SODA*, pages 1275–1284, 2007.

[28] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain, and G. Bishop-Hurley. Wireless adhoc sensor and actuator networks on the farm. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 492–499, 2006.

[29] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling with dynamic deadlines. *IEEE Transactions on Mobile Computing*, 6(4):395–410, 2007.

[30] R. Sugihara and R. K. Gupta. Speed control and scheduling of data mules in sensor networks. *ACM Transactions on Sensor Networks*, 7:1–29, 2010.

[31] H. Wang, K. Yao, G. Pottie, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *Proceedings of the 3rd international*

symposium on Information processing in sensor networks, IPSN '04, pages 36–45. ACM, 2004.

[32] B. Wietrzyk and M. Radenkovic. Energy efficiency in the mobile ad hoc networking approach to monitoring farm animals. In *Proceedings of the Sixth International Conference on Networking*. IEEE Computer Society, 2007.

[33] X. Wu, K. N. Brown, and C. J. Sreenan. Analysis of smartphone user mobility traces for opportunistic data collection in wireless sensor networks. *Pervasive and Mobile Computing*, July 2013.