Path Planning of Data Mules in Sensor Networks

RYO SUGIHARA and RAJESH K. GUPTA
University of California, San Diego

We study the problem of planning the motion of "data mules" for collecting the data from stationary sensor nodes in wireless sensor networks. Use of data mules significantly reduces energy consumption at sensor nodes compared to commonly-used multihop forwarding approaches, but has a drawback that it increases the latency of data delivery. Optimizing the motion of data mules, including path and speed, is critical for improving the data delivery latency and making the data mule approach more useful in practice. In this paper, we focus on the path selection problem: finding the optimal path of data mules so that the data delivery latency can be minimized. We formulate the path selection problem as a graph problem that is capable of expressing the benefit from larger communication range. The problem is NP-hard and we present approximation algorithms for both single data mule case and multiple data mules case. We further consider the case in which we have only partial knowledge of communication range, where we design semi-online algorithms that improves the offline plan using online knowledge at runtime. Simulation experiments on Matlab and ns2 demonstrate that our offline and semi-online algorithms produce significantly shorter path lengths and data delivery latency compared to previously proposed methods, suggesting that controlled mobility can be exploited much more effectively.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; G.2.2 [**Mathematics of Computing**]: Graph Theory—*Path and circuit problems*

General Terms: Algorithms, Design, Performance

Additional Key Words and Phrases: Controlled mobility, approximation algorithm, integer programming, semi-online algorithm, simulation

1. INTRODUCTION

Data mules, i.e., mobile devices that we can control the motion provide an alternative way for collecting data from spatially dispersed sensor nodes. Unlike data collection via multihop forwarding among the nodes, data mules travel across the sensing field and communicate with each node when it is in the proximity. Data mules have been used in recent sensor network applications, e.g., a robot in underwater environmental monitoring [Vasilescu et al. 2005] and a UAV (unmanned aerial vehicle) in structural health monitoring [Mascarenas et al. 2008]. A benefit in data mule approach is that, by eliminating the need for multihop forwarding of data, energy consumption at the nodes is significantly reduced. On the other hand, a drawback is an increased data delivery latency, which is mainly governed by the

This work has been partially supported by the DMEA research program under contract H94003-07-C-0702 and the US National Science Foundation under grants CCF-0702792 and SRS-0820034.

motion of data mules. Thus optimizing the motion of data mules is an important problem for data mule approach to be useful in practice.

Motion planning problem of data mules consists of three parts: determining the path, calculating speed changes over time, and scheduling when to collect data from each node. Although these are not independent from each other, simultaneously optimizing them is hard, as indicated by the NP-hardness of path selection problem alone, which we show later in the paper. To deal with this hardness, previous studies [Kansal et al. 2004; Somasundara et al. 2004; Ma and Yang 2006; Xing et al. 2007] simplified the problem by using simple mobility models and/or simple communication models. However, these simplifications lead to suboptimal solutions that incur unnecessarily large data delivery latency. To balance solution quality and problem tractability, we have proposed the data mule scheduling (DMS) problem as a unified problem framework for motion planning of data mules [Sugihara and Gupta 2007]. The main idea of the DMS framework is to increase the independence between the subproblems (path selection, speed control, and job scheduling) and optimize them mostly separately. The latter two subproblems correspond to one dimensional case of the problem (1-D DMS) where data mules move along the given paths.

In this paper we discuss the path selection problem. Following the idea of the DMS framework, we treat the path selection as an independent problem. Our strategy is to formulate the problem as a graph problem that we call the Label-Covering Tour (LCT) problem. We present approximation algorithms for the LCT problem for single data mule case and multiple data mules case. We also present integer linear program (ILP) formulation of the LCT problem for obtaining the lower bounds through LP relaxations. We demonstrate the performance of the approximation algorithms by simulation experiments, in which we compare our methods with previously proposed methods and with the lower bounds. We further discuss the case in which we only have partial knowledge about the communication range. For this case we present semi-online algorithms that make an initial plan only with offline knowledge and then update it at runtime to opportunistically exploit online information. Performance for this case is evaluated by using ns2 network simulator.

Our contributions are:

—Formulate the path selection problem of data mules as a graph problem and design approximation algorithms both for single and multiple data mules cases;

—Model the case of partially known communication range by proposing a novel hybrid connectivity model and design semi-online algorithms for the case;

—Demonstrate the effectiveness of the proposed algorithms by extensive simulation experiments on Matlab and ns2.

The rest of this paper is organized as follows. In Section 2 we introduce related work. In Section 3 we consider the path selection problem for single data mule case. We define the Label-Covering Tour problem and present an approximation algorithm, along with the results of simulation experiments. In Section 4 we discuss multiple data mules case. Besides an approximation algorithm, we also present an integer linear program (ILP) formulation of the problem and some experimental

results. In Section 5 we study the case of partially known communication range. We present semi-online algorithms and show the results of realistic simulation experiments using ns2. Finally Section 6 concludes the paper.

2. RELATED WORK

We mainly introduce related works on path selection problem in data mule approach. Refer to [Ekici et al. 2006] for overview of data mule and similar approaches that use controlled mobility in wireless sensor networks.

Path selection problem has been discussed in several different problem settings. Somasundara et al. [2004] studied the problem of choosing the path of a data mule that traverses through a sensor field with sensors generating data at a given rate. They designed heuristic algorithms to find a path that minimizes the buffer overflow at each sensor node. In their subsequent work [Somasundara et al. 2007], they presented a heuristic algorithm for multiple data mules case based on the formulation as a vehicle routing problem (VRP). Gu et al. [2006] presented an improved algorithm for the same problem settings as [Somasundara et al. 2004]. In these works, it is assumed that data mules need to go to the sensor node's exact location to collect data (i.e., no remote communication). This assumption facilitates TSP-like formulations of the problem and makes the path selection problem of a data mule similar to packet routing problem such as the one studied in [Meliou et al. 2006], where the authors discuss the optimal routing scheme of a query packet and the requested data. However, these formulations result in underutilized communication capability, since data mules can actually collect data from nodes without visiting their exact locations via wireless communications.

Zhao and Ammar [2003] studied the problem of optimally controlling the motion of a data mule in mobile ad-hoc networks. A data mule, which is called a "message ferry" in their work, mediates communications between sparsely deployed stationary nodes. They considered the remote communication, but path selection is done based on a TSP-like formulation. They extended their work to multiple data mules case in [Zhao et al. 2005] and presented heuristic algorithms. In our work, we realize a more optimized path selection where the data mule only needs to visit subset of nodes as long as it travels inside the communication range of each node at least once.

Xing et al. [2007] designed path selection algorithms when each node can forward data toward the base station along a routing tree constructed in advance. Their formulation is also similar to TSP and also assumed the existence of forwarding nodes that do not generate data by themselves, in order to make the network connected and enable the construction of routing tree rooted at the base station. Although these assumptions allow the fail-over mechanism that improves the data delivery rate, they also limit the applicability of the technique. Our problem framework can express not only their settings[1], but also more general settings including disconnected networks.

Ma and Yang [2006] discussed the path selection problem under different assumptions. Their objective is to maximize the network lifetime, which is defined as the

[1]Combination of data mule and multihop forwarding approaches is also expressible in the DMS problem framework. Please refer to [Sugihara and Gupta 2009] for more details.

time until the first node dies (i.e. minimum of the lifetime of all nodes). They considered the remote wireless communication and also multihop communication among nodes. When the path of data mule is given, they showed the problem of maximizing the network lifetime is formulated as a flow maximization problem that has a polynomial time algorithm. Choosing the path of data mule is done by their heuristic algorithm that uses the divide and conquer approach and finds a near-optimal path for each part of the nodes. A problem in their algorithm is that it is applicable only to special configurations in which the data mule starts from the left end of the area, travels toward the right end and then comes back to the left end. Hence, for example, it is not clear how to use the algorithm for circular area having the base station in the center. Our algorithms do not have any restrictions on the shape of node deployment area.

We study semi-online scheduling problem for the case when the communication range is known only partially. The term "semi-online scheduling" mainly appears in the context of job scheduling, and refers to the cases when partial information is available offline. Common examples of partial information are optimum makespan, order of job arrivals, etc. For more on these topics, see [Pruhs et al. 2004]. In our case, part of feasible intervals (i.e., intervals in which a job can be executed) is known in advance, but the job is possibly executable also in other intervals. This is not a standard assumption in the scheduling literature. It is partly because scheduling problems for jobs with multiple feasible intervals are not common. There are few papers discussing multiple feasible intervals for nonpreemptive case [Shih et al. 2003; Chen et al. 2005] and unit length intervals case [Simons and Sipser 1984], but neither is identical to our problem settings.

3. PATH SELECTION PROBLEM

In this section we discuss the path selection problem for single data mule case. First we introduce the data mule scheduling (DMS) problem as a problem framework for motion planning problem of data mules. Then we give a formal definition of the path selection problem. Our main idea is to formulate it as a graph problem, which we call the Label-Covering Tour problem. The Label-Covering Tour problem is to find a minimum-cost path that intersects with the communication ranges of all nodes, so that a data mule can collect data from them. We prove the problem is NP-hard and present an approximation algorithm. Further we evaluate the performance of the algorithm by simulation experiments.

3.1 Data Mule Scheduling (DMS) Problem Framework

Before going into details of the path selection problem, we introduce the data mule scheduling (DMS) problem framework for motion planning problem of data mules.

Motion planning of data mules is a hard problem. Communications with sensor nodes need to take place at the neighbor of each node and will take certain time duration, whereas the motion of data mule is possibly governed by dynamics constraints. There is also a prioritization problem when the data mule is in the communication ranges of multiple nodes.

To deal with this complexity, in the DMS framework, we decompose the problem into loosely connected subproblems. Specifically, as shown in Figure 1, we can decompose the problem into the following three subproblems:

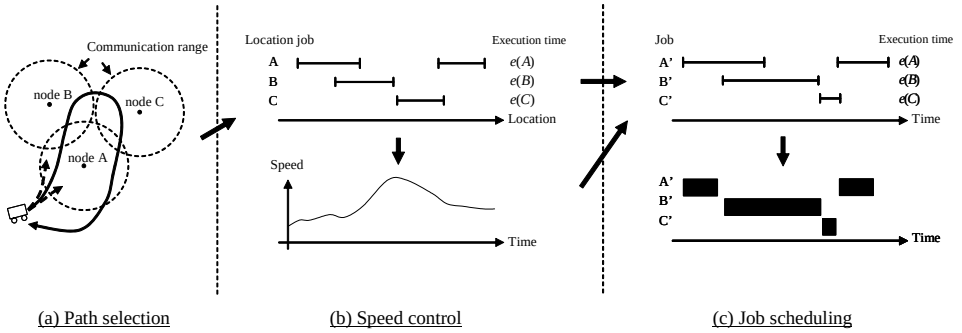

Fig. 1. Subproblems of data mule scheduling

(1) Path selection: which trajectory the data mule follows

(2) Speed control: how the data mule changes the speed while moving along the path

(3) Job scheduling: from which sensor the data mule collects data at each time point

Path selection is to determine the trajectory of the data mule in the sensor field. To collect data from each particular sensor, the data mule needs to go in the sensor's communication range at least once.

Speed control is the second subproblem to determine how the data mule changes its speed along the chosen path. The data mule needs to change the speed so that it stays within each sensor's communication range long enough to collect all the data from it.

The final subproblem is job scheduling. Once the time-speed profile is determined, we get a mapping from each location to a time point. Thus we get a scheduling problem by regarding data collection from each sensor as a job. Each job has one or more intervals in which it can be executed. Job scheduling is to determine the allocation of time to jobs so that all jobs can be completed.

In this paper, we focus on the path selection subproblem. The speed control and job scheduling subproblems are considered together as the one dimensional DMS (1-D DMS) problem, which is discussed in [Sugihara and Gupta 2010b]. The 1-D DMS problem applies to the cases in which data mules need to move along fixed paths.

The DMS problem framework is general and can be used to express several earlier problems in the area. For instance, the assumption of no remote wireless communication (as in [Somasundara et al. 2004; 2007]) is easily expressed by setting the communication range to zero in the path selection subproblem. The constant speed assumption (as in [Ma and Yang 2006; 2007; Xing et al. 2007]) and variable speed assumption (as in [Zhao and Ammar 2003; Kansal et al. 2004]) are handled in the speed control subproblem.

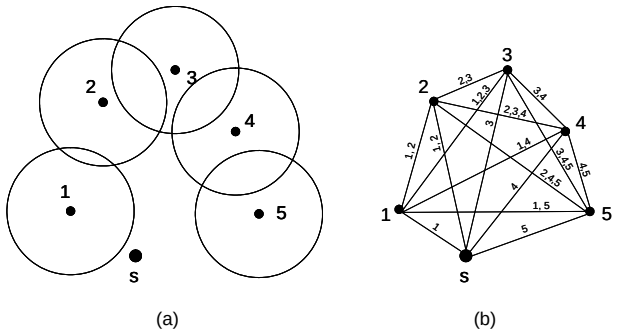

Fig. 2. Simplifying the path selection problem using a labeled graph representation: (a) Instance of path selection problem. (b) Corresponding labeled graph.

3.2 Problem Description

The ultimate objective of the path selection problem is to find a path such that the shortest travel time (= latency) can be realized in the corresponding 1-D DMS problem. However, it is not clear which path results in shorter travel time. For example, even if the path length is short, the travel time would be long if the intersections of the path and communication range of each node are short, because the data mule needs to slow down to collect all the data. Moreover, it is also difficult to search an optimal path in a brute-force manner when the data mule can freely move around within the space.

To deal with these issues, we simplify the path selection problem. To reduce the solution space, we consider a complete graph having vertices at sensor nodes' locations and assume the data mule moves between vertices along a straight line. Each edge is associated with a cost and a set of labels, where the latter represents the set of nodes whose communication ranges intersect with this edge. In other words, the data mule can collect data from these nodes while traveling along this edge. We want to find a minimum-cost tour that the data mule can collect data from all the nodes. We discuss later how we assign the cost to each edge so that a tour with smaller cost results in shorter travel time.

Figure 2 is an example that depicts the basic idea of the formulation. Figure 2(a) shows five nodes and their communication ranges, in addition to the base station (shown as "s"), where the data mule starts and brings the data back. From this input, we construct a labeled undirected complete graph as shown in Figure 2(b). Each edge e has a set of labels $L(e) \subseteq L$ and cost $c(e)$, where $L = \{l_1, ..., l_n\}$ is the set of all labels and n is the number of sensor nodes. We determine $L(e)$ as follows: $l_i \in L(e)$ if node i's communication range intersects with edge e. Intuitively, by moving along edge e, the data mule can collect data from the nodes whose labels are in $L(e)$.

Now we define the Label-Covering Tour problem formally as follows: Given an undirected complete graph $G = (V, E)$ where each vertex in $V = \{v_0, v_1, ..., v_n\}$ is a point in $\mathbf{R}^2$, a cost function on edges $c : E \rightarrow \mathbf{Q}_{\mathbf{0}}^{+}$, a set $L = \{l_1, ..., l_n\}$ of labels, and a set of constants $\{r_1, ..., r_n\}$. Each edge $e_{ij} \in E$ is associated with

subset $L_{ij} \subseteq L$. For $k = 1, ..., n$, $l_k \in L_{ij}$ iff the Euclidean distance between v_k and an edge e_{ij} is equal to or less than r_k. A tour T is a list of points that starts and ends with v_0, allowing multiple visits to each point. A tour T is "label-covering" when it satisfies at least one of the following conditions for $k = 1, ..., n$: 1) $\exists e_{ij} \in T(E), l_k \in L_{ij}$, where $T(E)$ is the set of edges traversed by T, or 2) $dist(v_0, v_k) \leq r_k$, where $dist(v_i, v_j)$ is the Euclidean distance between v_i and v_j. Find a label-covering tour T that minimizes the total cost $\sum_{e_{ij} \in T(E)} c_{ij}$.

Unfortunately, this simplified problem is still NP-hard.

THEOREM 3.1. *Label-Covering Tour is NP-hard.*

PROOF. We show metric TSP is a special case of Label-Covering Tour. First we choose the cost function c to satisfy the triangle inequality (e.g., Euclidean distance). For a given set of points $V = \{v_0, ..., v_n\}$, by choosing small r_i's, we can make $dist(v_0, v_i) > r_i$ for all $i > 0$, $L_{ij} = \{l_i, l_j\}$ for all $i, j > 0$, and $L_{0j} = \{l_j\}$ for all $j > 0$. For such r_i's, any label-covering tour must visit all the points. An optimal label-covering tour does not visit any point multiple times except v_0 at the start and the end of the tour, since in such cases, we can construct another label-covering tour with smaller total cost by "shortcutting." Therefore, an optimal label-covering tour is an optimal TSP tour for V. □

As the cost metric c, we use Euclidean distance in the rest of the paper. As mentioned earlier, there is no rigorous argument that it is always an appropriate metric in the sense that minimizing the total cost leads to minimum data delivery latency. Nevertheless we choose Euclidean distance based on the empirical fact that, when we formulate the path selection problem as the Label-Covering Tour problem, there is a high positive correlation between total Euclidean distance and the travel time of data mule. See [Sugihara and Gupta 2008] for more details.

The Label-Covering Tour problem is merely one possible way of formulating the path selection problem. Clearly TSP is also an option, as used in many studies, though it has a problem that we cannot leverage the communication range. An alternative formulation that takes account the communication range is TSP with Neighborhoods (TSPN) problem, in which the problem is to find a shortest tour that visits each of given regions instead of points. Application of TSPN problem to path selection is proposed in some recent studies [Yuan et al. 2007; Tekdas et al. 2009]. In the evaluation section we compare the performance by the Label-Covering Tour formulation and TSPN formulation.

The formulation as the Label-Covering Tour problem can easily be extended in several ways to deal with different problem settings. For example, suppose there are obstacles in the field that do not allow a data mule to move from one node to another along a straight line. We can consider this case by eliminating some edges from the graph G, which is otherwise a complete graph. Another example is to allow a data mule to make turns at the locations other than the nodes' locations, so that we can consider more flexible paths. This is also useful for the above obstructed case. We can incorporate this just by adding more vertices in the graph.

3.3 Approximation Algorithm

We design an approximation algorithm for the Label-Covering Tour problem. As discussed earlier, Euclidean distance is used as the cost metric. This enables us

—Make a TSP tour T using an exact or approximation algorithm for metric TSP
—Construct a directed graph (V, E) where $V = \{T(i)\}$ and $E = \{T(i)T(j)|$ for all l, $i \leq l \leq j$, Euclidean distance between line segment $T(i)T(j)$ and $T(l)$ is equal to or less than $r\}$.
—Find shortest path from $T(0)$ to $T(n)$ (e.g., by Dijkstra's algorithm).

Fig. 3. Approximation algorithm for Label-Covering Tour: $T(i)$ is the i-th vertex that the tour T visits. $T(0)$ is the starting vertex.

to design an approximation algorithm by using known algorithms for metric TSP where the triangle inequality holds.

Figure 3 shows the approximation algorithm for Label-Covering Tour. It first finds a TSP tour T by using any algorithm (exact or approximate) for TSP. Then it finds a short[2] label-covering tour that is obtained by shortcutting T.

Computation time of the algorithm is $\mathcal{C}_{TSP} + O(n^3)$, where $\mathcal{C}_{TSP}$ denotes the computation time of the algorithm used for solving TSP. If we add a constraint $j \leq i + k$ for some constant k in the definition of E in the second step of the algorithm, the computation time becomes $\mathcal{C}_{TSP} + O(nk^2)$, which is useful for large n when the above cubic-time computation is not feasible. An additional benefit about the algorithm related to computation time is that it does not require all edges to be labeled beforehand. Especially when the number of nodes is large, making the labeled graph itself may take a long time proportional to $O(n^3)$. The proposed approximation algorithm computes the label of an edge as necessary when it makes a decision on whether to skip a node.

Next we analyze the approximation factor of the algorithm. Let T_{OPT}, T_{APP} denote the optimal label-covering tour and the approximate label-covering tour, respectively. Total length of tour T is denoted as $|T|$. Also let α be the approximation factor of the TSP algorithm used in the first step of the approximation algorithm. Then we have the following theorem:

THEOREM 3.2. $|T_{APP}| \leq \alpha(|T_{OPT}| + 2nr)$

PROOF. Clearly $|T_{APP}| \leq \alpha|T_{TSP}|$, where T_{TSP} is the optimal TSP tour. We give a lower bound to T_{OPT} by constructing another TSP tour by modifying T_{OPT}. Figure 4 shows the idea of construction. The points A and B (shown in filled circles) are visited by T_{OPT} and other points in the figure (shown in non-filled circles) are not. We call the former "visited points" and the latter "non-visited points". By the definition of a label-covering tour, any non-visited points are within distance r from either a traversed edge or a visited point of a label-covering tour. For example in the figure, all of AC and DD', ..., GG' have the length less than r. Then we can construct a "tour"[3] that is identical to T_{OPT} but takes a detour to visit each non-visited point (e.g., ACAD'DD'...B). Since there are at most n non-visited points,

[2]In [Sugihara and Gupta 2008], we have misstated that the algorithm finds the shortest label-covering tour. There is no guarantee that the label-covering tour obtained by the algorithm is the shortest one of all the shortcut tours of T.
[3]This is not a tour in our definition because it does not consist of edges between the nodes.

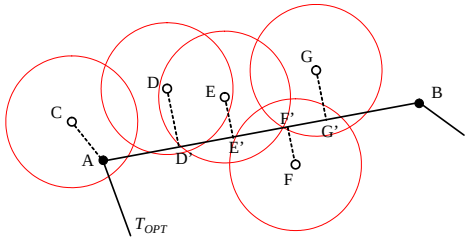

Fig. 4. Constructing a TSP tour from the optimal label-covering tour T_{OPT}: every non-visited point is within distance r from T_{OPT}.

total length of detour is at most $2nr$. This "tour" is easily converted to a shorter TSP tour by skipping all additional points (e.g., D', E', ...) and apply shortcutting so that each point is visited exactly once. Therefore, we have $|T_{OPT}|+2nr \geq |T_{TSP}|$. The theorem follows by combining this and $|T_{APP}| \leq \alpha|T_{TSP}|$. □

3.4 Performance Evaluation

We evaluate the performance of the approximation algorithm by numerical experiments. We have implemented the algorithm in Matlab.

3.4.1 *Method.* We deploy n nodes in the circular area of radius d that has the base station at the center. The nodes are randomly placed within the circle. Each of the nodes has a circular communication range of identical radius r. For each edge connecting a pair of nodes, we assign a set of labels by calculating the distance from the line segment and each node. We use Concorde TSP solver[4] to find an optimal TSP tour.

Using the tour, we transform the original problem to 1-D DMS problem and calculate the travel time by using the heuristic algorithm presented in [Sugihara and Gupta 2010a]. We assume that each node has the same execution time $e = 10[sec]$ and also that the speed of data mule needs to be zero at each point where it changes the direction[5].

For each (n, d, r), we generate 50 node deployments and take the average for the results. For 1-D DMS, we use the maximum absolute acceleration $a_{max} = 1[m/s^2]$, and the maximum speed $v_{max} = 10[m/s]$ to roughly simulate the motion of a helicopter as in [Mascarenas et al. 2008].

Figure 5 shows some examples of label-covering tours for a node deployment with different communication ranges. As the communication range grows, the number of visited points becomes less and the path length becomes shorter.

3.4.2 *Effect of node density and network size.* Figure 6(a) shows the relation between the communication range and the total travel time for different node density. To see how the size of communication range affects the travel time, we have normalized the total travel time by the one when the communication range is zero.

[4] `http://www.tsp.gatech.edu/concorde/index.html`

[5] Otherwise, it would require infinite acceleration, since we assume a path consists of only line segments and not curves.

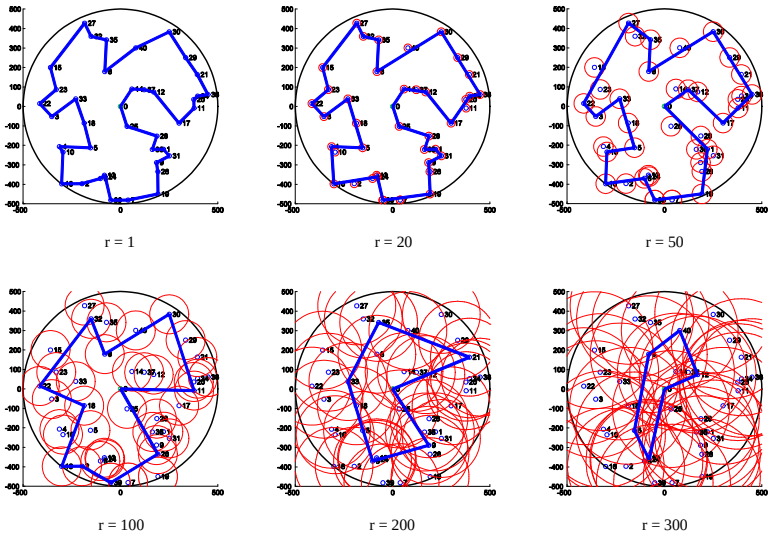

Fig. 5. Label-covering tours for different communication ranges: 40 nodes, $d = 500[m]$; Path of data mule is shown in bold line.

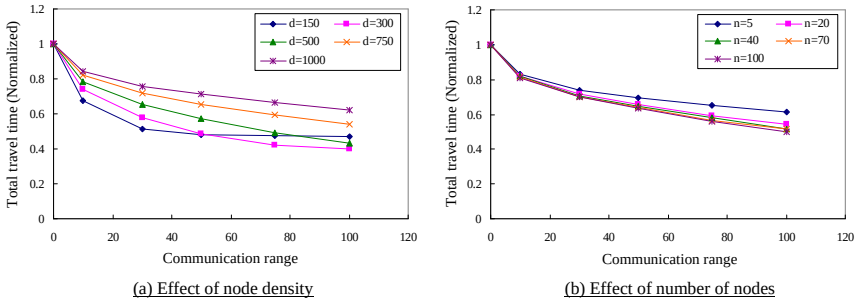

Fig. 6. Comparison of total travel time for (a) different node density (40 nodes) and (b) different number of nodes ($d = 500[m]$ for 20 nodes): $a_{max} = 1[m/s^2]$, $v_{max} = 10[m/s]$

The graph shows that the total travel time is reduced in all cases by up to 50% for this parameter set, suggesting the proposed problem formulation and algorithm altogether successfully exploit the breadth of communication range. The amount of reduction is bigger when the density is higher (i.e., smaller d), except the case of $d = 150[m]$ for large communication ranges. This is because the total travel time is already very close to the absolute lower bound, which is the product of the execution time and the number of nodes.

Figure 6(b) shows the effect of number of nodes, varied from $n = 5$ to $n = 100$. We set d to $500[m]$ when $n = 20$, and changed d in proportion to $\sqrt{n}$ so that the density remains constant. The results show the reduction of total travel time for large communication ranges, but no big difference for different number of nodes.

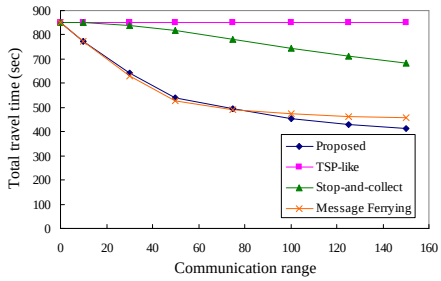

Fig. 7. Comparison of total travel time for different path selection algorithms: 40 nodes, $d = 500[m]$, $a_{max} = +\infty$, $v_{max} = 10[m/s]$

3.4.3 *Comparison with other strategies.* Next we compare the travel time of our approximation algorithm with those of previously proposed motion planning algorithms as listed below.

—TSP-like: Based on the model used in [Somasundara et al. 2004]. Data mule visits all nodes. It stops at each node location to collect data and moves to the next node. While moving, the speed is constant at v_{max}. We use optimal TSP tours.

—Stop-and-collect: Based on the model used in [Ma and Yang 2007]. Data mule takes a label-covering tour, as in our approximation algorithm. However, it stops to collect data when it is in the communication range of each node. While moving, speed is constant at v_{max}. We find tours by using our approximation algorithm with optimal TSP tours[6].

—Message Ferrying: Based on the algorithm proposed in [Zhao and Ammar 2003]. Data mule visits all nodes as in the TSP-like algorithm, but communication is also done while moving. Speed is variable between 0 and v_{max}. Speed and data collection schedule are determined by solving a linear program such that the total travel time is minimized. We use optimal TSP tours.

To allow direct comparison, we set $a_{max} = +\infty$ for our proposed approximation algorithm, since all other algorithms assume data mule can change its speed instantly. Note that when $a_{max} = +\infty$, we can obtain an exact solution for the 1-D DMS problem by solving a linear program [Sugihara and Gupta 2010b].

Figure 7 shows the results for a representative case for 40 nodes. When the communication range is small, the travel time does not differ among the algorithms. As the communication range grows, Message Ferrying and the proposed algorithm show larger improvements than other two methods, and the proposed algorithm gets gradually better than Message Ferrying. When the communication range is $150[m]$, the proposed algorithm is nearly 10% better than Message Ferrying, 40% better than Stop-and-collect, and more than 50% better than TSP-like method.

[6]We could not use the path selection algorithm proposed in [Ma and Yang 2007], since it has a restriction on the configuration of data mule and deployment area. Specifically, it assumes the data mule starts from the left end of the deployment area, travels toward the right end, and comes back to the initial position.

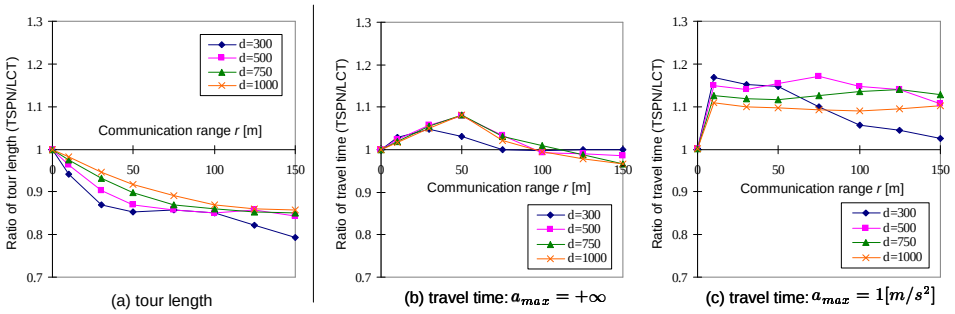

Fig. 8. Comparison of Label-Covering Tour and TSPN formulations (40 nodes): (a) ratio of tour length; (b) ratio of travel time: $a = +\infty$, (c) $a = 1[m/s^2]$

When there is an acceleration constraint (i.e., $a_{max} \neq +\infty$), which none of the previous studies has addressed, the gaps between the proposed algorithm and others are expected to be larger. This is because all of these methods require the data mule to stop more frequently than the proposed algorithm does.

3.4.4 *Comparison with TSPN-based formulation.* As discussed in Section 3.2, Label-Covering Tour is not a sole option for formulating the path selection problem of a data mule. One of the alternative formulations that has been proposed in the literature [Yuan et al. 2007; Tekdas et al. 2009] is as a TSPN (TSP with Neighborhoods) problem. We compare the travel time for these formulations.

Since TSPN is an NP-hard problem, we use an approximation algorithm presented in [Elbassioni et al. 2005] (called "Algorithm $\mathcal{A}$" in the paper). We chose this algorithm because it always produces a polygonal path, whereas some algorithms (e.g., [Dumitrescu and Mitchell 2003]) consider paths with arcs [7]. The algorithm assumes each region may have different diameter and works as follows:

(1) Order the points by their diameter $\delta_1 \leq \delta_2 \leq ... \leq \delta_n$.
(2) Pick the point p_1 on the smallest region randomly. For $i = 2...n$, pick the point p_i on the i-th region that minimizes $\min_{1 \leq j \leq i-1} dist(p_i, p_j)$.
(3) Construct a TSP tour on $\{p_1, ..., p_n\}$ using any TSP algorithm.

For our case, we choose the base station as the first point ("p_1") in the second step and use optimal TSP tours for the last step.

Figure 8 shows the comparison of results for two algorithms based on different formulations. As Figure 8(a) shows, the TSPN-based formulation always produces shorter tours than the LCT-based one. However, as Figure 8(b) shows, travel time is equal or longer in the TSPN-based formulation in many cases. This is more prominent in the acceleration constrained case shown in Figure 8(c), where the travel time is longer by more than 10% in most cases.

Figure 9 explains the reason of these results. In this example, though the LCT tour is longer than the TSPN tour, it contains much fewer number of turns. This

[7]Note that, when the path includes arcs, we need to consider angular velocity and acceleration, and thus we cannot treat path selection and speed control problems separately anymore.

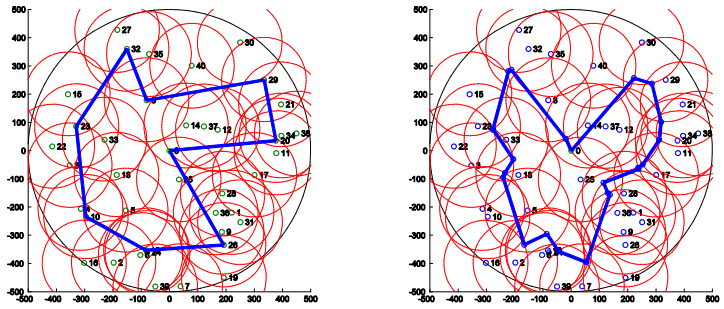

Fig. 9. Example of tours for same node deployment (40 nodes, $d = 500[m]$, $r = 150[m]$): Label-covering tour contains 8 turns, while TSPN tour contains 23 turns.

is reasonable since in the approximation algorithm for LCT problem, we effectively reduce the number of edges in the tour by skipping vertices in a TSP tour. When there is an acceleration constraint (Figure 8(c)), the data mule needs to stop at each turn and thus more frequent stops tend to make the travel time longer due to acceleration and deceleration. When there is no acceleration constraint (i.e., $a = +\infty$), the number of turns itself does not affect the travel time. However, since a TSPN tour tends to intersect the communication range of each node only at a point, the data mule needs to stop at the point to collect data from the node. As a result, the travel time for a TSPN tour is not as good as expected from its tour length.

These arguments on the advantage of the LCT-based formulation over the TSPN-based one are not theoretically supported[8]. Certainly there are the cases that the TSPN-based formulation produces better performance and moreover there are a number of algorithms for finding TSPN tours other than the one we used. Nevertheless, it is experimentally shown for realistic parameters that the LCT-based formulation often outperforms the TSPN-based one and the results can be explained with appropriate reasons. With some additional reasons such as that fewer turns with long edges make it easier for the data mule to maneuver, we claim that the LCT-based formulation, along with the proposed approximation algorithm, has a good balance of performance and practicality.

4. MULTIPLE DATA MULES CASE

In this section, we first define the path selection problem for multiple data mules case and then present an approximation algorithm. We also present an integer linear program (ILP) formulation of the problem and apply relaxations in several ways to obtain the lower bounds. In the end we present the results from simulation experiments.

[8]As an additional reason to prefer the LCT-based formulation, it can easily handle the case with obstacles in the field (mentioned in the end of Section 3.2), while it is not immediately clear in the TSPN-based ones.

4.1 Problem Definition

Based on the Label-Covering Tour problem[9] for single data mule case, we define k-Label-Covering Tour (k-LCT) problem for k data mules case as follows. We are given an undirected complete graph $G = (V, E)$ where each vertex in $V = \{v_0, v_1, ..., v_n\}$ is a point in $\mathbf{R}^2$, a cost function on edges $c : E \to \mathbf{Q}_\mathbf{0}^+$, a set $L = \{l_1, ..., l_n\}$ of labels, and constants $\{r_1, ..., r_n\} \in \mathbf{Q}_\mathbf{0}^+$ and $K \in \mathbf{Z}^+$. Each edge $e_{ij} \in E$ is associated with subset $L_{ij} \subseteq L$. For $p = 1, ..., n$, $l_p \in L_{ij}$ iff the Euclidean distance between v_p and edge e_{ij} is equal to or less than r_p. A subtour T is a list of subset of all vertices that starts and ends with v_0, allowing multiple visits to each vertex. A set of subtours $\{T_1, T_2, ..., T_K\}$ is "label-covering" when it satisfies at least one of the following conditions for $p = 1, ..., n$: 1) $\exists k, e_{ij} \in T_k(E), l_p \in L_{ij}$, where $T_k(E)$ is the set of edges traversed by T_k, or 2) $dist(v_0, v_p) \leq r_p$, where $dist(v_i, v_j)$ is the Euclidean distance between v_i and v_j. Find a set of label-covering subtours $\{T_1, T_2, ..., T_K\}$ that minimizes the maximum of cost of subtours $\max_k \sum_{e_{ij} \in T_k(E)} c(e_{ij})$.

As in the previous section, we focus on the case that c is the Euclidean distance.

4.2 Approximation Algorithm

Our strategy in designing an approximation algorithm is to solve TSP for k salesmen (k-TSP) first and then to shortcut each subtour so that the label-covering property is maintained. For solving k-TSP problem, we use k-SPLITOUR algorithm [Frederickson et al. 1978]. k-SPLITOUR algorithm constructs k subtours by splitting 1-TSP tour in the following way:

—Find a 1-TSP tour $R = (v_0, v_1, ..., v_n, v_0)$ with $\sum_{e \in R(E)} c(e) = D$.

—For each $j, 1 \leq j < k$, find the last vertex $v_{p(j)}$ such that the cost of the path from v_0 to $v_{p(j)}$ along R is not greater than $(j/k)(D - 2c_{max}) + c_{max}$, where $c_{max} = \max_i c(e_{0i})$.

—Form k subtours as $R_1 = (v_0, v_1, ..., v_{p(1)}, v_0), R_2 = (v_0, v_{p(1)+1}, ..., v_{p(2)}, v_0)$, $..., R_k = (v_0, v_{p(k-1)+1}, ..., v_n, v_0)$.

We have the following theorem about the approximation ratio:

THEOREM 4.1 [FREDERICKSON ET AL. 1978]. *If $\hat{C}_k$ is the cost of the largest of the k subtours generated by k-SPLITOUR algorithm, and C_k^* is the cost of the largest subtour in an optimal solution of k-TSP, then*

$$\hat{C}_k / C_k^* \leq \alpha + 1 - 1/k,$$

where α is the bound for the single traveling salesman algorithm.

Figure 10 shows the algorithm for the k-LCT problem. In the algorithm, we first use k-SPLITOUR algorithm to construct k subtours. Then we apply shortcutting for each subtour as long as the label-covering property is not violated. Shortcutting is attempted on the longest subtour first. If not successful, we try the second longest one, and the third one, etc., until there is no subtours that can be shortcutted. We have the following guarantee on the approximation ratio:

[9]We call it the "1-LCT" problem for clarity in the rest of the paper.

—Find k-TSP subtours $\{R_1, R_2, ..., R_k\}$ using k-SPLITOUR algorithm.
—While true,
  —For each subtour R, in the decreasing order of total cost,
    —Find a visited vertex v s.t.
      —the label-covering property is maintained if v is skipped by R, and
      —$c(R) - c(R\backslash v)$ is maximized.
    —If v is found, $R \leftarrow R\backslash v$ and break the inner loop; Otherwise continue.
  —If no vertex was skipped in all subtours in this iteration, stop and output the subtours.

Fig. 10. Approximation algorithm for k-LCT problem

THEOREM 4.2. *If APP is the approximate solution of a given instance of the k-LCT problem and OPT is the optimal one,* $APP \leq (\alpha + 1 - \frac{1}{k})(OPT + 2nr)$, *where* α *is the approximation ratio of TSP algorithm and* r *is the communication range.*

PROOF. As we construct k-label-covering subtours by shortcutting k-TSP subtours, we have $APP \leq \hat{C}_k$. By definition of the label-covering property, for any unvisited vertex there exists an edge in k-label-covering subtour within distance c. Thus, by using the same technique for Theorem 3.2 to convert a label-covering tour into a TSP tour, we have $C_k^* \leq OPT + 2nr$. Then, from Theorem 4.1, the theorem follows. □

4.3 Integer Linear Program Formulations

First we give an ILP formulation of the 1-LCT problem. Variables are

—$x_{ij} \in \{0, 1\}$: edge $e_{ij} = (v_i, v_j)$ is included in the tour iff $x_{ij} = 1$
—$y_i \in \{0, 1\}$: node v_i is visited iff $y_i = 1$

Constants are

—$c_{ij} \in \mathbf{Q}_0^+$: cost of edge e_{ij}
—$d_{i,pq} \in \{0, 1\}$: equals 1 iff node v_i is within the distance r_i from edge e_{pq}.

Then the 1-LCT problem is
Minimize $\sum_{i,j} c_{ij} x_{ij}$
Subject to

$$x_{ii} = 0 \ (\forall i) \quad , \quad y_0 = 1 \tag{1}$$

$$y_i \leq \sum_j x_{ji} \ = \ \sum_j x_{ij} \leq (n-1) y_i \quad \forall i \tag{2}$$

$$\sum_{p,q} x_{pq} d_{i,pq} \ \geq \ 1 + y_i \quad \forall i \geq 1 \tag{3}$$

$$\frac{1}{|S|} \sum_{i \in S} y_i \leq \sum_{i \in S, j \notin S} x_{ij} \ \leq \ \sum_{i \in S} y_i \quad \forall S \subseteq V \backslash \{v_0\} \tag{4}$$

$$\frac{1}{|S|} \sum_{i \in S} y_i \leq \sum_{i \in S, j \notin S} x_{ji} \ \leq \ \sum_{i \in S} y_i \quad \forall S \subseteq V \backslash \{v_0\} \tag{5}$$

Inequality (2) enforces in- and out-degree of each vertex to be equal. It also enforces that both degrees are zero when the vertex is not visited. We obtain this by combining the following two constraints:

—if $y_i = 0$, $\sum_j x_{ji} = \sum_j x_{ij} = 0$
—if $y_i = 1$, $\sum_j x_{ji} = \sum_j x_{ij}$

Inequality (3) is the label-covering property. It is obtained by combining the following two constraints:

—if $y_i = 0$, $\sum_{p,q} x_{pq} d_{i,pq} \geq 1$
—if $y_i = 1$, no constraint ($\sum_{p,q} x_{pq} d_{i,pq} \geq 2$ is trivially satisfied)

Finally, inequalities (4) and (5) are the constraints for eliminating invalid subtours. They are obtained by combining the following two constraints:

—if $\sum_{i \in S} y_i = 0$, $\sum_{i \in S, j \notin S} x_{ij} = \sum_{i \in S, j \notin S} x_{ji} = 0$
—if $\sum_{i \in S} y_i > 0$, $\sum_{i \in S, j \notin S} x_{ij} \geq 1$ and $\sum_{i \in S, j \notin S} x_{ji} \geq 1$

Note that these subtour constraints consist of exponential number of inequalities. However, we can use the cutting-plane technique (e.g., [Pataki 2003]), in which we solve the ILP problem without these constraints first, add only violated inequalities and solve again, and repeat this until we obtain the tour without invalid subtours.

We can easily extend this formulation to the k-LCT problem in the following way. Instead of x_{ij}, y_i, the variables are $x_{ij}^{(k)}$ and $y_i^{(k)}$, representing whether edge e_{ij} is included in k-th tour (i.e., tour of k-th data mule) and whether vertex v_i is visited by k-th tour, respectively. To allow the min-max objective, we have an additional variable z. Constants are the same as in the 1-LCT problem. Then, k-LCT problem is to minimize z subject to $\forall k. \sum_{i,j} c_{ij} x_{ij}^{(k)} \leq z$ and inequalities (1)-(5), with substituting $x_{ij}^{(k)}$, $y_i^{(k)}$ for x_{ij}, y_i.

4.4 Obtaining Lower Bounds

The ILP problem above cannot be solved in a realistic time when the number of variables is large. Instead, we use that for obtaining lower bounds of the optimal solution by applying various relaxations.

Lower bounds are useful because of the following reason. In Theorem 4.2, we have obtained a theoretical guarantee on the performance of the approximation algorithm for the k-LCT problem. However, the upper bound given in the theorem is loose when r is large. This gives a motivation for evaluation by experiments, in which we compare the approximate solution with lower bounds to figure out the performance in practical settings.

We consider the following relaxations to obtain the lower bounds:

—**ILPcover**: ILP without subtour constraints: This finds subtours that collectively satisfy the label-covering property. Note that the number of subtours is arbitrary and some of them may not include v_0.
—**LPCP**: LP relaxation + cutting plane: first solve LP relaxation of the original ILP with the subtour constraints for $|S| = 2$ case. Then, cutting plane method is applied.

—**MaxCost**: Trivial lower bound by $2\max\{\max_i\{c_{0i}\} - r, 0\}$. This is the smallest possible cost to touch the communication range of the farthest node.

We need a different approach to use cutting plane method in **LPCP**. In the original ILP formulation, we just needed to find invalid subtours in an intermediate solution, add subtour constraints for them, and repeat that until we find a valid solution. However, this does not work in **LPCP** because intermediate solutions are generally fractional in the LP relaxation and thus cannot identify invalid subtours directly. Instead, we regard the value of $x_{ij}^{(k)}$ as the weight of edge (i, j) and add the subtour constraints for the cycles with large mean weight.

Finding a cycle that has the maximum mean weight in a graph is done in polynomial time [Karp 1978]. We modify the algorithm to find only the cycles with length at least three and applied it iteratively by eliminating the vertices in the cycle. Subtour constraints are added if they were not added previously. If no new invalid subtours were found, or it reached the maximum number of iterations (set to 10), the solution at that point is used.

The following theorem enables us to obtain a lower bound for k-DM case by "scaling" the result for 1-DM case:

THEOREM 4.3. *For a given graph G and a lower bound LB_1 of the 1-LCT problem for G, $LB_1/k \leq OPT_k$, where OPT_k is the optimal solution for the k-LCT problem for G.*

PROOF. From any set of k-LCT subtours, by connecting each subtour, we can make a 1-LCT tour. If there exists a set of k-LCT subtours whose maximum length is strictly less than OPT_1/k, the 1-LCT tour made by connecting these subtours have length strictly less than OPT_1, which is a contradiction. Therefore we have $LB_1/k \leq OPT_1/k \leq OPT_k$. □

4.5 Performance Evaluation

We evaluate the performance of the approximation algorithm in Figure 10 in realistic situations by simulation experiments. We first compare the path length by the proposed algorithm with two other strategies and also with the lower bounds. Then, we compare the travel time in these strategies by solving the 1-D DMS problem.

4.5.1 *Method.* We use Matlab for simulation. For simulations, nodes are deployed at random locations in a circular area with the base station at the center. Number of nodes is 40 and radius of deployment area is fixed to $d = 600[m]$. We compute our results as the average of ten different node deployments. The number of data mules is $k = \{1, 2, 3, 4\}$ and communication range is $r = \{0, 50, 100, 150\}[m]$. Euclidean distance is used as the cost function c. We use Concorde TSP solver to find an optimal TSP tour. For the 1-D DMS problem, we set maximum acceleration of data mule a_{max} to $1[m/s^2]$ and maximum speed v_{max} to $10[m/s]$. Execution time is $e = \{10, 30, 60\}[sec]$ for each node. We use the heuristic algorithm [Sugihara and Gupta 2010a] to solve the 1-D DMS problem under acceleration constraint.

We have implemented the following three strategies for comparison:

—**Proposed**: Use k-LCT subtours obtained by the approximation algorithm (Figure 10).

—**Overlay**: Use a 1-TSP tour for all k data mules and each data mule collects equal amount of data from each node, i.e., symmetric schedule on identical paths. This models the SIRA (single-route algorithm) strategy in [Zhao et al. 2005].

—**Partition**: Use k-TSP subtours and each data mule collects data only from the nodes on the subtour it is assigned. This models the MURA (multi-route algorithm) strategy in [Zhao et al. 2005].

4.5.2 *Results.* Figure 11 shows the maximum path length for different number of data mules k and different size of communication range r. When k is changed, the path length does not change for the overlay strategy, since it always uses 1-TSP tours regardless of k. Both of the partition strategy and the proposed strategy scale well with the number of data mules. When the size of communication range r is changed, the path length did not change in either the overlay or partition strategies, because both of them use (k-)TSP tours. In the proposed strategy, the length decreased for larger communication range.

In comparing with the lower bounds, the average ratios of approximate solutions to the lower bounds are less than two in the tested cases. We cannot guarantee anything from these results, since we do not know how tight the lower bounds are and how bad the ratio can be in other parameters. Nevertheless, this is useful information to give estimates to the practical performance, since the bound by Theorem 4.2 is very loose in many of the tested cases.

Figure 12 compares the maximum travel time of the three strategies. We have tested three different execution time e to see the effect of the communication bandwidth and/or the amount of data in each sensor node. In $e = 10$ case, the proposed strategy yielded up to $30 - 40\%$ shorter travel time than other two strategies. As e increases, which corresponds to less communication bandwidth or larger amount of data in sensor nodes, the differences between the strategies shrink. This is because the execution time becomes more dominant in determining the travel time than the path length.

5. PATH SELECTION UNDER PARTIALLY KNOWN COMMUNICATION RANGE

The algorithms we have presented so far are based upon the assumption that the communication range is entirely known. In this section, we relax this assumption and assume the range is known only partially, which simulates realistic wireless environments more closely. We first present hybrid connectivity model that consists of known and unknown communication ranges. Then we present semi-online algorithms that work under the hybrid connectivity model by combining offline scheduling and opportunistic improvement at runtime. We use ns2 network simulator[10] for performance evaluation.

5.1 Preliminaries

5.1.1 *Connectivity Model.* A simple connectivity model is the circular, fixed-range (i.e., "Unit Disk") model. In this model, received signal strength at distance is estimated by considering mean path loss, and the radius of the circle is determined by thresholding the strength. Despite its appearance in many papers, it has been

[10] `http://www.isi.edu/nsnam/ns/`

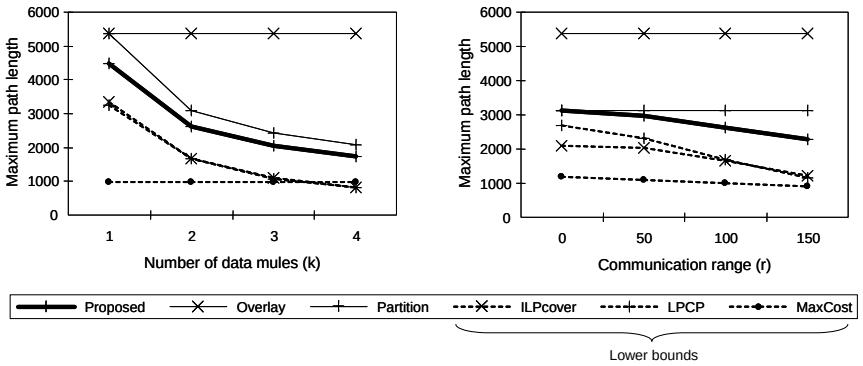

Fig. 11. Maximum path length: (left) For different k. Communication range is fixed to $r = 100$; (right) For different r. Number of data mules is $k = 2$. Data for ILPcover ($k = 2, 3, 4$) and LPCP ($k = 3, 4$) are due to the scaling of $k = 1$ case.

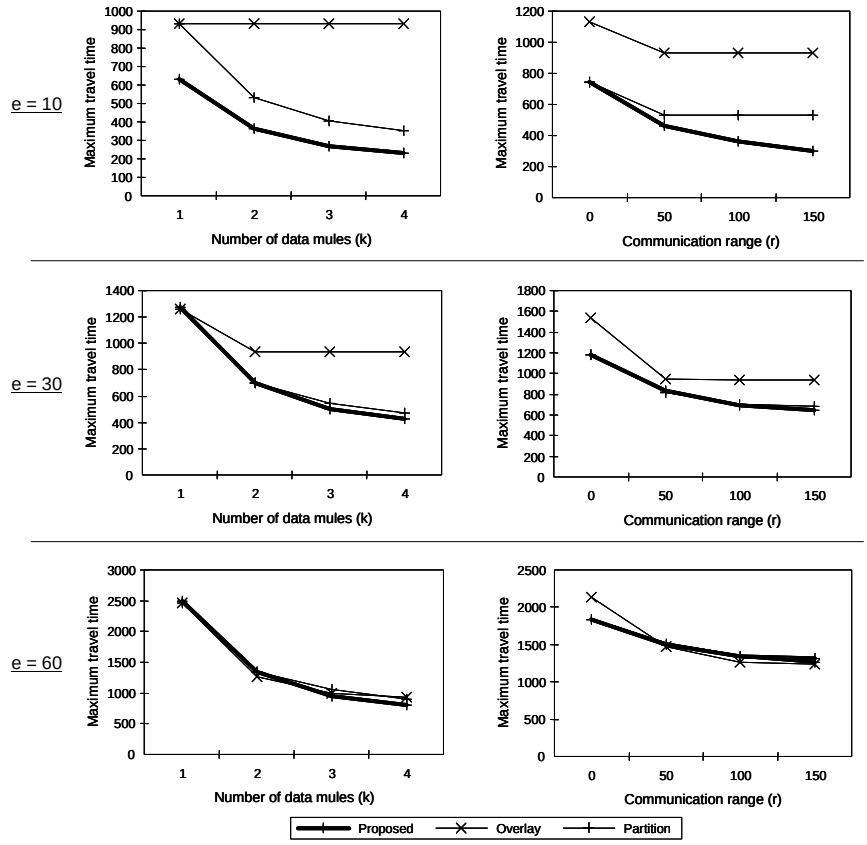

Fig. 12. Maximum travel time when execution time $e = \{10, 30, 60\}$. (left) For different k. Communication range is fixed to $r = 100$; (right) For different r. Number of data mules is $k = 2$.

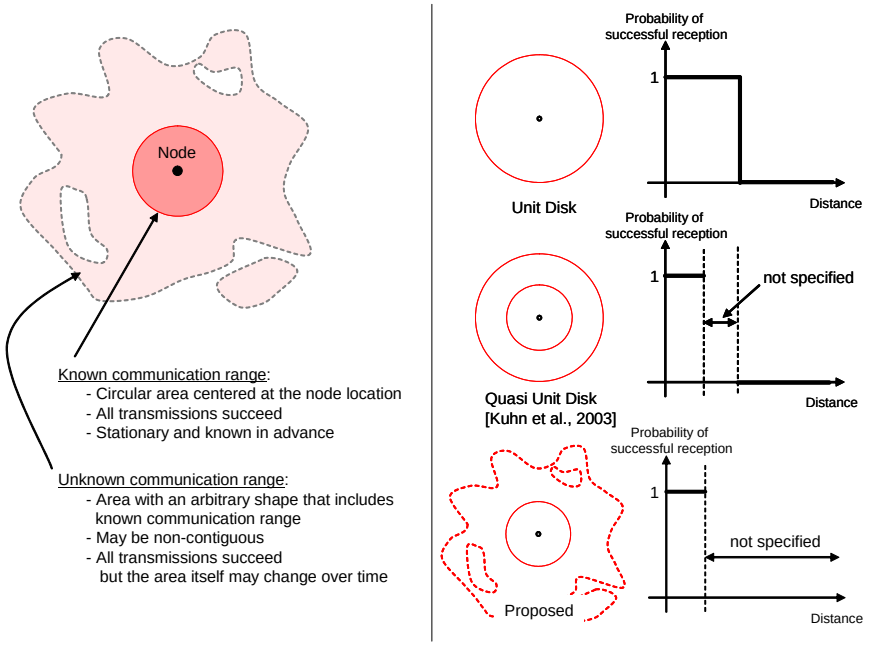

Fig. 13. Hybrid connectivity model (left) and comparison with other models (right)

pointed out that this model does not reflect the reality of the wireless channel [Ganesan et al. 2002; Kotz et al. 2003].

In contrast, probabilistic models takes shadowing and fading into account. In these models, there is always a finite probability of communication failure. In combinatorial frameworks used for solving the path and scheduling problems, this results in excessively conservative assumptions such as "communication is possible only at the exact location of each node" as in [Somasundara et al. 2007; Xing et al. 2008] or "no knowledge about connectivity" as in [Somasundara et al. 2006]. Furthermore, a recent study shows empirically that probabilistic models are still insufficient to model many realistic aspects [Lee et al. 2007]. Specifically, it is pointed out in [Lee et al. 2007] that, in the Shadowing model implemented in ns2, noises are assumed spatially independent and follow Gaussian, both of which are not true in real environments.

Our hybrid connectivity model is based on the observation that, even though the connectivity at distance fluctuates dramatically, there is a certain range in the vicinity of node that we can guarantee the connectivity for sure. Figure 13 shows the idea of the hybrid connectivity model. We have a fixed circular range called *known communication range* around the node. Communication is always successful in the known communication range. The size of known communication range needs to be chosen accordingly, depending on the degree of uncertainty of channel and the environments (indoor/outdoor, terrain, etc.).

A larger range that contains known communication range is *unknown communication range*. Although communication is always successful also in the unknown

communication range, the shape is unknown and may change over time. This represents the uncertainty at distance, and because of this, it is not known in advance if communication at a point outside of known communication range of a node will succeed or not at certain time. We do not make any assumptions about the deterministic or probabilistic characteristics of the unknown communication range, except that it always contains the known communication range. This is the biggest difference from Quasi Unit Disk model [Kuhn et al. 2003] that assumes an inner circle just like the known communication range in our model, but also an outer circle to limit the maximum connectivity range.

Having the known communication range enables an offline algorithm that has a performance guarantee. On the other hand, the performance is largely affected by the size of communication range, as shown earlier in the paper. Thus the opportunity that an offline algorithm misses becomes huge under more uncertain environments, because there is a larger portion of unknown communication range that offline algorithms cannot leverage. This motivates us to consider semi-online algorithms, which are based on offline plans based on partial knowledge and try to improve at runtime with new information.

5.1.2 *Assumptions and Notations.* The radius r_K of known communication range is same for all nodes and is known. The unknown communication range is not known offline, but the data mule can tell whether its current location is in the unknown communication range of each node (i.e., whether the data mule can collect data from that node)[11]. There is only one data mule. The data mule knows its current location and moves along a polygonal path starting and ending at the base station. The data mule can change the speed in the range of $[0, v_{max}]$ without any acceleration constraint.

We use P_{Off} and S_{Off} to denote the path and schedule by the offline algorithm, respectively. A schedule is represented as a set of schedule entry $s = (I, job, v)$, where $s(I)$ is the location interval on the path, $s(job)$ is the job to execute (i.e., node to collect data from), and $s(v)$ is the speed of data mule. The travel time by schedule S_{Off} is denoted as T_{Off}.

In addition to the non-periodic case that a data mule travels the field only once, we also consider the periodic case. In the periodic case of the problem, each node continuously generates data at a given rate and the data mule travels the field periodically to keep collecting the data and bringing them back to the base station. We assume that the data mule needs to stop at the base station for a constant time T_b in the end of each period. This is to account for the time for depositing all the data to the base station, refueling the data mule, and so on.

5.2 Semi-Online Algorithms

We first present a semi-online algorithm for the non-periodic case, in which sensor nodes have a fixed amount of data in advance and a data mule travels the field once to collect all the data. Then we extend the algorithm for the periodic case, in which sensor nodes generate data at a certain rate and a data mule periodically

[11]This is a strong assumption and will be removed when we design a communication protocol and do simulations in ns2.

Algorithm 1 Semi-online algorithm

```
1: S ← S_Off                                          ▷ Copy offline schedule
2: repeat
3:     if ∃s ∈ S. x ∈ s(I) then                       ▷ x: current location
4:         v ← s(v)
5:         execute s(job)
6:         if x = end of s(I) then
7:             S ← S\s
8:         end if
9:     else
10:        v ← v_max
11:        Execute any available jobs
12:    end if
13:    Eliminate all s ∈ S s.t. s(job) is finished
14:    if ∃s ∈ S. x ∈ s(I) then
15:        dest ← end of s(I)
16:    else if S ≠ ∅ then
17:        dest ← start of next(S)          ▷ Start of the next schedule entry in S
18:    else
19:        dest ← BS                                   ▷ BS: base station
20:    end if
21: until arrive at the BS
```

travels the field.

5.2.1 *Non-Periodic Case.* Algorithm 1 shows the pseudocode for the semi-online algorithm. At first the data mule follows the offline path and schedule, and does opportunistic data collection when there is no schedule entry to execute at the current location (Lines 3-13). When one of the jobs finishes, the data mule takes a shortcut to the location where the next schedule entry starts, moving at the maximum speed (Lines 16-20). Figure 14 explains this. In the figure, the bold lines (A-B-C, D-E) on the offline path P_{Off} represent the intervals covered by offline schedule entries. Now, in the semi-online algorithm, assume that the data collection from Node 1 finished at P, due to that some of the data have been collected beforehand by opportunistic data collection. Then, the schedule entries covering A-B-C are removed and the next schedule entry to execute is the one covering D-E. So the data mule sets its destination to D and takes a shortcut path P-D. On P-D, the data mule moves at the maximum speed and does opportunistic data collection. Since the length of P-D is shorter than that of P-B-D and the speed is maximum, we can further reduce the travel time.

The following theorem guarantees that the travel time is shorter than that of the offline algorithm:

THEOREM 5.1. $T_S \leq T_{Off}$

PROOF. Let $\mathcal{I}_{idle}, \mathcal{I}_{busy}$ denote the sets of location intervals that the data mule is idle and busy in the offline schedule S_{Off}, respectively.

Consider another offline schedule S'_{Off} that is exactly the same in all of $\mathcal{I}_{busy}$

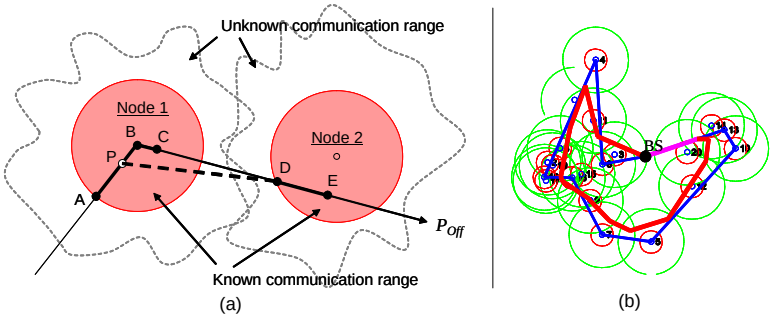

Fig. 14. Semi-online algorithm: (a) If the data collection from Node 1 finishes at P, the data mule directly heads for D, where the next offline schedule entry starts; (b) Example of path by the semi-online algorithm: path by offline schedule is also shown.

but takes a shortcut between the intervals in $\mathcal{I}_{busy}$, where the data mule moves at the maximum speed. Let T'_{Off} denote the travel time for S'_{Off}, then $T'_{Off} \leq T_{Off}$ clearly holds.

In the semi-online algorithm, when the data mule does not fully cover one of the intervals in $\mathcal{I}_{busy}$, it means that the job has finished before reaching the finishing location as planned in S_{Off}. Since the data mule can directly head to the start of the next schedule entry, it may further reduce[12] the total travel length from S'_{Off}. As movements outside of $\mathcal{I}_{busy}$ are always at the maximum speed, we have $T_S \leq T'_{Off}$ and the theorem follows. □

5.2.2 *Periodic Case.* We can use the algorithm for the non-periodic case to design the one for the periodic case. The main change from the non-periodic case will be in deciding whether to skip a schedule entry or not, because the data is continuously generated and in a sense, a job can never be finished. Our idea is to use a simple strategy: the data mule simply tries to collect from each node as much data as possible according to the offline schedule: i.e., replace "$s(job)$ is finished" in Line 13 of Algorithm 1 with "node $s(job)$ is empty." When a node's buffer once becomes empty, the node is regarded as "finished" and the schedule entries for that node are skipped throughout the current period.

We can construct a periodic offline algorithm in the following way. We use the same path selection algorithm described in Figure 3 and solve the 1-D DMS problem for the periodic case. When there is no acceleration constraint, the periodic 1-D DMS problem is solved optimally either by a closed formula or by linear programming [Sugihara and Gupta 2010b].

The following two lemmas state that the periodic semi-online algorithm collects more data in shorter amount of time compared to the offline algorithm.

LEMMA 5.2. *Let $T^{(k)}_{Sp}, T_{Off}$ denote the travel time in the k-th period for the periodic semi-online algorithm and the periodic offline algorithm. Then, for all k, $T^{(k)}_{Sp} \leq T_{Off}$.*

[12]Travel length is unchanged when the interval, the new destination, and the current location are aligned on a line.

PROOF. Immediately follows from Theorem 5.1. □

LEMMA 5.3. *Let $c_i^{(k)}, c_i^{Off}$ denote the amount of data collected from i-th node in the periodic semi-online algorithm (in k-th period) and in the periodic offline algorithm, respectively. Then, for any k and all i, either $c_i^{(k)} \geq c_i^{Off}$ or i-th node's buffer becomes empty during k-th period.*

PROOF. If i-th node does not become empty during k-th period, all schedule entries for i-th node are fully executed by the semi-online algorithm. Thus, at least c_i^{Off} is collected. In addition, the semi-online algorithm may collect more data opportunistically. □

Then the following theorem guarantees that the system is stable; i.e., the amount of data in each node does not increase indefinitely:

THEOREM 5.4. *Assume there exists a feasible offline schedule for a given set of nodes, data generation rate λ_i, stop time T_b, and v_{max}. After sufficiently large number of periods, for the periodic semi-online algorithm, the amount of data in each node is less than some constant.*

PROOF. For sufficiently large k, we show that the amount of data in each node is not increasing. We consider the following three cases, classified by the travel time and the amount of collected data: (i) $T_{Sp}^{(k)} = T_{Off}$ and $c_i^{(k)} = c_i^{Off}$ for all i, (ii) $T_{Sp}^{(k)} = T_{Off}$ and $\exists i.c_i^{(k)} \neq c_i^{Off}$, and (iii) $T_{Sp}^{(k)} < T_{Off}$. From Lemma 5.2, these three cases enumerate all possibilities. In Case (i), for any node, the amount of data generated and collected are the same. Thus the amount of data in each node is not increasing. In Case (ii), for i's that satisfy $c_i^{(k)} \neq c_i^{Off}$, by Lemma 5.3, either $c_i^{(k)} > c_i^{Off}$ or i-th node becomes empty during k-th period. For the first case, the amount of data in the node will decrease. For the second case, the amount is less than c_i^{Off}, which is a constant. Finally in Case (iii), the theorem holds for the nodes that become empty for the same reason as above. For other nodes, from Lemma 5.3, the data mule collects more than c_i^{Off} in the time strictly less than T_{Off}. Thus the amount of data in each node will decrease. □

5.3 Communication Protocol

Let us consider how the data mule communicates with each node in either offline-scheduled or opportunistic data collection. We design a simple request-response-based communication protocol. In this protocol, communications are always initiated by the data mule. This helps keep the implementation simple at the nodes, which have only limited computational resources.

Note that the protocol design described here is one of the simplest examples. We can possibly improve the throughput in several ways, for example by letting nodes send multiple packets per single request and introducing a windowing scheme as in TCP.

5.3.1 *Basic Operation.* Data from nodes is sent and acknowledged packet by packet in the following way. The data mule sends a request packet that includes a request ID and requested data size. The data mule keeps track of the latest request ID for each node. When a node receives a request, it responds to the data mule by

sending the data of the requested size with attaching the request ID. When the data mule receives a packet with the newest request ID for that node, it increments the request ID and sends the next request. If the data mule does not receive a response for a request within the predetermined timeout period, it regards the packet was lost and sends the previous request again.

5.3.2 *Scheduled Data Collection.* For the scheduled data collections in the offline plan, the data mule sends a request only to the node designated in each schedule entry. It continues to send requests until one of the following events happens:

(1) Amount of collected data reached the size designated in the schedule entry.

(2) The data mule arrived at the endpoint of the schedule entry.

(3) The buffer of the node became empty.

Ideally the first and second events happen at the same time, but they usually do not, due to the error in estimating the effective bandwidth.

5.3.3 *Opportunistic Data Collection.* For the opportunistic data collection, a practical issue is that the data mule cannot tell whether it is in the unknown communication range of a node without actually communicating with it. This can be accomplished by using advertisement packets. When the data mule tries to start opportunistic data collection, it first broadcasts an advertise packet. When a node receives an advertise packet, it responds to that by sending the size of data in its buffer to the data mule. Then, when the data mule receives the response packet, it sends a request to that node just as in the scheduled data collection. In this way, the data mule can communicate with multiple available nodes in parallel and can adapt dynamically to transient connectivity. To find new nodes in range, the data mule issues advertisement packets periodically during opportunistic data collection.

5.4 Simulation Experiments

To evaluate the effect of introducing the hybrid connectivity model and the benefit of using semi-online algorithms in realistic radio environments, we conduct simulation experiments in ns2 with the Shadowing propagation model.

5.4.1 *Methods.* We have implemented the offline algorithm and the semi-online algorithm for the DMS problem in the periodic data generation case. We implemented the offline algorithm in Matlab and generated a Tcl script for ns2. The semi-online algorithm along with the communication protocol are implemented as the modules of ns2. We run the script on ns2 version 2.33.

To assess the performance, we measure the delivery latency for each data packet from the time it is generated until the time the base station receives it. For each test case, the simulation is repeated multiple periods until it reaches stability. We consider it stable when the average delivery latency of the data received in the current period is within $\pm 1\%$ of that of the previous period. If it is stable, we use the data for the next period as the final results.

We use the Shadowing propagation model in ns2. In the Shadowing model, the received power $P_r(d)$ at distance d is derived as the ratio to that at reference

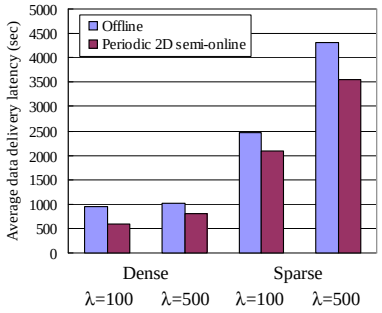

Fig. 15. Simulation results: Average data delivery latency: 50 nodes, average of 10 experiments for each case.

distance d_0 as follows:

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta \log\left(\frac{d}{d_0}\right) + X_{dB}$$

where β is path loss exponent, X_{dB} is a Gaussian random variable with zero mean and standard deviation σ_{dB}, which is called the shadowing deviation. Based on [Stuedi et al. 2005], we set $d_0 = 1.0, (\beta, \sigma_{dB}) = (3.0, 6.0)$ to simulate outdoor environments. We set the size of known communication range r_K to $20[m]$, where the theoretical successful reception probability is 99.9%.

Other parameters are as follows. Fifty nodes are randomly placed in a circular area of radius $200[m]$ (dense) or $500[m]$ (sparse). We generate 10 node deployments for each. For all deployments, the base station is placed at the center of the circular area and the data mule starts from and comes back to the base station. Data generation rate λ at each node is 100 or $500[Bytes/sec]$. In the communication protocol, the request timeout is 200 msec and the period to issue advertisement packet is 5 sec. In ns2, we use 802.11 MAC with RTS/CTS and bandwidth 2 Mbps. Packet size is set to 400 Bytes. We determined the effective bandwidth by a simple experiment: the data mule and a node are placed 10 m apart and, using the communication protocol above, the data mule tries to collect data as much as possible within 10 sec. The average of 10 measurements with different seeds for random number generator of ns2 was 402440 Bytes, which corresponds to 322.0 Kbps. Based on these results, we use 320 Kbps as the effective bandwidth.

5.4.2 *Main Results.* Figure 15 shows the average data delivery latency for each of the four deployment cases. All tested cases reached the stability condition. Average number of periods until getting stable was 4.0 (min: 4, max: 4) for the offline algorithm and 5.3 (min: 4, max: 8) for the semi-online algorithm, respectively. The average data delivery latency was lower in the semi-online algorithm in all cases. The decrease was larger in the dense deployments (38.9% and 20.6% for λ =100 and 500, respectively) than in the sparse deployments (15.4% and 17.4% for λ =100 and 500, respectively).

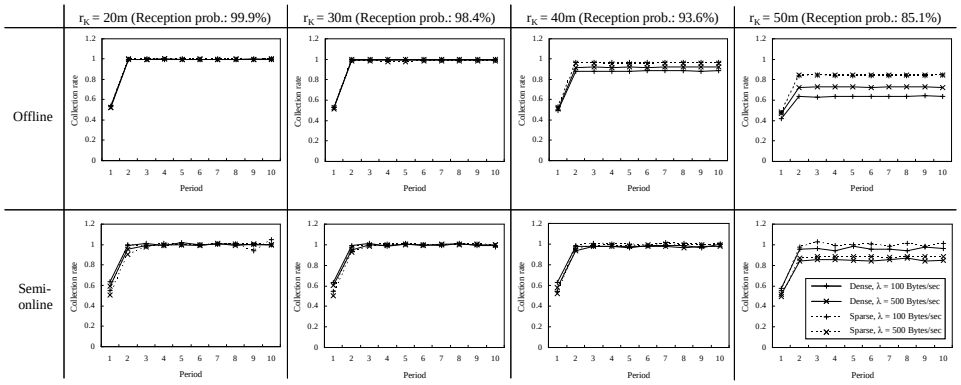

Fig. 16. Simulation results: Effect of overestimating r_K (size of known communication range).

5.4.3 *Effects of Inaccurate Parameters.* For the hybrid connectivity model and the algorithms (both offline and semi-online) to work, we need to estimate two parameters: the size of known communication range (r_K) and the effective bandwidth. Larger r_K implies better performance, but it is not clear about the consequences when it is larger than the reality. We have a similar issue for effective bandwidth, too, especially when the actual effective bandwidth fluctuates over time. Here we see the effects of overestimating these parameters on both the offline and the semi-online algorithms.

We use "collection rate" as the performance metric for these experiments. Collection rate R_C is calculated in each period and, for k-th period, it is defined as follows:

$$R_C^{(k)} = \frac{\sum_i c_i^{(k)}}{\sum_i g_i^{(k)}},$$

where $g_i^{(k)}$ is the amount of data generated at i-th node in k-th period and $c_i^{(k)}$ is the amount of data collected by the data mule in k-th period. For the data to be collected without any loss, R_C needs to be 1 on average. On the other hand, if R_C is constantly lower than 1, data accumulates at each node over time and eventually overflows, resulting in loss of data.

As in the previous experiments, we test each case on four deployments: combinations of two node densities (dense and sparse) and two data generation rates ($\lambda = 100$ and 500). For the experiments on known communication range, we test on $r_k = 20$ (default), 30, 40, and 50. Theoretical probabilities of successful reception for these values are 99.9%, 98.4%, 93.6%, and 85.1%, respectively. For the experiments on effective bandwidth, we test on 320 Kbps (default), 480 Kbps, and 640 Kbps.

Figure 16 shows the effect of overestimating r_K. In the offline algorithm, the average of R_C for the deployments (from period 6 to 10) was 1.000, 0.991, 0.933, 0.762 for r_K =20, 30, 40, 50, respectively. The average was higher in the semi-online algorithm: 1.000, 0.999, 0.989, and 0.923, respectively.

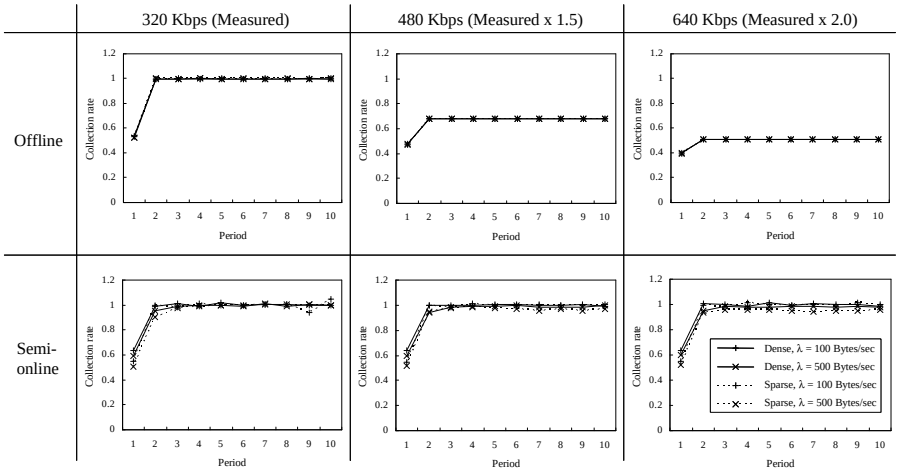

Fig. 17. Simulation results: Effect of overestimating the effective bandwidth.

Figure 17 shows the effect of overestimating the effective bandwidth. The average of R_C for the deployments (from period 6 to 10) was 1.000, 0.679, 0.509 for the 320 Kbps case, 480 Kbps case, 640 Kbps case, respectively. These values almost agree with the ratio to the actual effective bandwidth ($\approx$ 322 Kbps). For the semi-online algorithm, the average of R_C was much higher: 1.000, 0.989, 0.983 for the effective bandwidth of 320 Kbps, 480 Kbps, 640 Kbps, respectively.

To summarize, the experiments with overestimated r_K and effective bandwidth showed that, in these cases, we can achieve higher collection rate in the semi-online algorithm than in the offline algorithm. This suggests that the semi-online algorithm is beneficial in terms of the robustness against inaccurate parameters, as well as the performance improvements as demonstrated in the first experiments.

6. CONCLUSIONS

When using data mules in sensor networks for data collection, path selection has a large impact on the latency of data delivery from when it is generated at a sensor node until it is delivered to the base station. To find a short path with a reasonable amount of computation, we have formulated the path selection problem as the Label-Covering Tour (LCT) problem, in which we find the shortest path that visits a subset of nodes and intersects with the communication ranges of all nodes. We have shown that the LCT problem is NP-hard and designed an approximation algorithm. We have also extended the problem and algorithm for the multiple data mules case, where we gave an integer linear program formulation to obtain lower bounds of the solution. Simulation experiments have demonstrated our formulation and approximation algorithms successfully exploit large communication range and perform better than previous methods. Finally we have considered the case in which the communication range is only partially known. We have designed the semi-online algorithms for this case and implemented them on ns2 network simulator. Compared to the offline algorithms, the semi-online algorithms are shown to produce

consistently better results and provide more robustness against uncertainty, both of which are preferable for real deployments.

REFERENCES

Chen, J.-J., Wu, J., Shih, C., and Kuo, T.-W. 2005. Approximation algorithms for scheduling multiple feasible interval jobs. In *RTCSA '05: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 11–16.

Dumitrescu, A. and Mitchell, J. S. B. 2003. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms 48,* 1, 135–159.

Ekici, E., Gu, Y., and Bozdag, D. 2006. Mobility-based communication in wireless sensor networks. *IEEE Communications Magazine 44,* 7 (July), 56–62.

Elbassioni, K. M., Fishkin, A. V., Mustafa, N. H., and Sitters, R. 2005. Approximation algorithms for euclidean group tsp. In *ICALP '05: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*. 1115–1126.

Frederickson, G. N., Hecht, M. S., and Kim, C. E. 1978. Approximation algorithms for some routing problems. *SIAM Journal on Computing 7,* 2, 178–193.

Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., and Wicker, S. 2002. Complex behavior at scale: An experimental study of low-power wireless sensor networks. *UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013*.

Gu, Y., Bozdağ, D., Brewer, R. W., and Ekici, E. 2006. Data harvesting with mobile elements in wireless sensor networks. *Computer Networks 50,* 17, 3449–3465.

Kansal, A., Somasundara, A. A., Jea, D. D., Srivastava, M. B., and Estrin, D. 2004. Intelligent fluid infrastructure for embedded networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. 111–124.

Karp, R. M. 1978. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics 23*, 309–311.

Kotz, D., Newport, C., and Elliott, C. 2003. The mistaken axioms of wireless-network research. *Dartmouth College Computer Science Technical Report TR2003-467*.

Kuhn, F., Wattenhofer, R., and Zollinger, A. 2003. Ad-hoc networks beyond unit disk graphs. In *DIALM-POMC '03: Proceedings of the 2003 joint workshop on Foundations of mobile computing*. 69–78.

Lee, H., Cerpa, A., and Levis, P. 2007. Improving wireless simulation through noise modeling. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*. 21–30.

Ma, M. and Yang, Y. 2006. SenCar: An energy efficient data gathering mechanism for large scale multihop sensor networks. In *DCOSS '06: Proceedings of the 2nd international conference on Distributed Computing in sensor systems*. 498–513.

Ma, M. and Yang, Y. 2007. SenCar: An energy efficient data gathering mechanism for large-scale multihop sensor networks. *IEEE Transactions on Parallel and Distributed System 18,* 10, 1476–1488.

Mascarenas, D., Flynn, E., Lin, K., Farinholt, K., Park, G., Gupta, R., Todd, M., and Farrar, C. 2008. Demonstration of a roving-host wireless sensor network for rapid assessment monitoring of structural health. W. Ecke, K. J. Peters, and N. G. Meyendorf, Eds. *Smart Sensor Phenomena, Technology, Networks, and Systems SPIE 6933,* 1, 69330K.

Meliou, A., Chu, D., Hellerstein, J., Guestrin, C., and Hong, W. 2006. Data gathering tours in sensor networks. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*. 43–50.

Pataki, G. 2003. Teaching integer programming formulations using the traveling salesman problem. *SIAM Review 45,* 1, 116–123.

Pruhs, K., Sgall, J., and Torng, E. 2004. Online scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Y.-T. Leung, Ed. CRC Press.

Shih, C., Liu, J. W. S., and Cheong, I. K. 2003. Scheduling jobs with multiple feasible intervals. In *RTCSA '03: Proceedings of the 9th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. 53–71.

Simons, B. and Sipser, M. 1984. On scheduling unit-length jobs with multiple release time/deadline intervals. *Operations Research 32,* 1, 80–88.

Somasundara, A. A., Kansal, A., Jea, D. D., Estrin, D., and Srivastava, M. B. 2006. Controllably mobile infrastructure for low energy embedded networks. *IEEE Transactions on Mobile Computing 5,* 8, 958–973.

Somasundara, A. A., Ramamoorthy, A., and Srivastava, M. B. 2004. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *RTSS '04: Proceedings of the 25th IEEE international Real-Time Systems Symposium*. 296–305.

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

Stuedi, P., Chinellato, O., and Alonso, G. 2005. Connectivity in the presence of shadowing in 802.11 ad hoc networks. In *WCNC'05: Proceedings of IEEE Wireless Communications and Networking Conference*. Vol. 4. 2225–2230.

Sugihara, R. and Gupta, R. K. 2007. Scheduling under location and time constraints for data collection in sensor networks. In *RTSS '07: Proceedings of the 28th IEEE international Real-Time Systems Symposium (work-in-progress session)*.

Sugihara, R. and Gupta, R. K. 2008. Improving the data delivery latency in sensor networks with controlled mobility. In *DCOSS '08: Proceedings of the 4th international conference on Distributed Computing in sensor systems*.

Sugihara, R. and Gupta, R. K. 2009. Optimizing energy-latency trade-off in sensor networks with controlled mobility. In *INFOCOM '09 (Mini-conference): Proceedings of the 28th Annual Joint Conference of the IEEE Computer and Communications Societies*. 2566–2570.

Sugihara, R. and Gupta, R. K. 2010a. Optimal speed control of mobile node for data collection in sensor networks. *IEEE Transactions on Mobile Computing 9,* 1, 127–139.

Sugihara, R. and Gupta, R. K. 2010b. Speed control and scheduling of data mules in sensor networks. *ACM Transactions on Sensor Networks 7,* 1.

Tekdas, O., Lim, J. H., Terzis, A., and Isler, V. 2009. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications 16,* 1, 22–28.

Vasilescu, I., Kotay, K., Rus, D., Dunbabin, M., and Corke, P. I. 2005. Data collection, storage, and retrieval with an underwater sensor network. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*. 154–165.

Xing, G., Wang, T., Jia, W., and Li, M. 2008. Rendezvous design algorithms for wireless sensor networks with a mobile base station. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. 231–240.

Xing, G., Wang, T., Xie, Z., and Jia, W. 2007. Rendezvous planning in mobility-assisted wireless sensor networks. In *RTSS '07: Proceedings of the 28th IEEE international Real-Time Systems Symposium*. 311–320.

Yuan, B., Orlowska, M., and Sadiq, S. 2007. On the optimal robot routing problem in wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering 19,* 9, 1252–1261.

Zhao, W. and Ammar, M. 2003. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proceedings of the IEEE Workshop on Future Trends in Distributed Computing Systems*. 308–314.

Zhao, W., Ammar, M., and Zegura, E. 2005. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. 1407–1418.

Received May 2009; revised Feb 2010, May 2010; accepted Aug 2010