Using Data Mules for Sensor Network Data Recovery

Jon Crowcroft[a], Liron Levin[b], Michael Segal[b]

[a] *Computer Laboratory, Cambridge University, Cambridge, United Kingdom. Email:* `Jon.Crowcroft@cl.cam.ac.uk`
[b] *Communication Systems Engineering Department, Ben-Gurion University of the Negev, Israel. Email:* `{levinlir,segal}@bgu.ac.il`

Abstract

In this paper, we study the problem of efficient data recovery using the data mules approach, where a set of mobile sensors with advanced mobility capabilities re-acquire lost data by visiting the neighbors of failed sensors, thereby avoiding permanent data loss in the network. Our approach involves defining the optimal communication graph and mules' placements such that the overall traveling time and distance is minimized regardless to which sensors crashed. We explore this problem under different practical network topologies such as arbitrary graphs, grids and random linear networks and provide approximation algorithms based on multiple combinatorial techniques. Simulation experiments demonstrate that our algorithms outperform various competitive solutions for different network models, and that they are applicable for practical scenarios.

1. Problem formulation

A data mule is a vehicle that physically carries a computer with storage between remote locations to effectively create a data communication link [21]. In ad-hoc networks, data mules are usually used for data collection [5] or monitoring purposes [11] when the network topology is sparse or when communication ability is limited. In this paper, we propose to extend the usage of data mules to the critical task of network reliability. That is, using the advantages of mobility capabilities to prevent losing crucial information while taking into consideration the additional operational costs. We propose to model the penalty of a sensor crash as the cost of restoring its information loss, and present several algorithms that minimize the total cost given any combination of failures. We use concepts from graph theory to model the deployment of the ad-hoc network and give special attention to linear and grid graph models, whose unique network characteristics makes them well suited for many sensor applications such as monitoring of international borders, roads, rivers, as well as oil, gas, and water pipeline infrastructures [13, 11].

Let T be a data gathering tree rooted at root r spanning n wireless sensors positioned in the Euclidean plane, where data propagates from leaf nodes to r. We model the environment as a complete directed graph $G = (V, E)$, where the node set represents the wireless sensors and the edge represents distance or time to travel between that sensors. We assume the sensors are deployed in rough geographic terrain with severe climatic conditions, which may cause sporadic failures of sensors. Clearly, if a sensor v fails, it is undesirable to lose the data it collected from its children in T, $\delta(v, T)$. Thus, a group of data gathering mules must travel through $\delta(v, T)$ and restore the lost information. We define this problem as (α, β)-Mule problem, where α is the number of simultaneous node failures and β is the number of traveling mules. For $\alpha = 1, \beta = 1$, the mule visits the children of v over the shortest tour, $t(m, \delta(v, T))$, starting and ending at node $m \in V$,

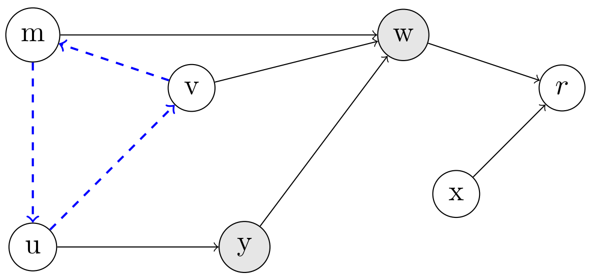


Figure 1: Example for the mule tour when 2 nodes fail. The grey nodes represent sensors that experienced failure and the blue dashed lines represent the mule tour; the tour starts and ends at node m.

where the length of the tour is equal to the Euclidean length of distances; the goal is to find a data gathering tree T, the placement of the mule m, and the shortest tours, $t(m, \delta(v, T))$ for all $v \in V$, which minimize the total traveling distance given **any** sensor can fail. Formally, find T and m such that $\sum_{v \in V} |t(m, \delta(v, T))|$ is minimized. In a similar way, we can define the problem for $\alpha > 1, \beta = 1$ (see example for $\alpha = 2$ in Figure 1, where the edges are directed towards the root). Formally, find T and m such that $\sum_{\{F \subset V : |F| = \alpha\}} |t(m, \bigcup_{v \in F} \delta(v, T))|$ is minimized. We can extend this scenario to the case where instead of a single mule, we have β mules $\bar{m} = \{m_1, m_2, ..., m_\beta\}$ deployed at different coordinates of the graph. When a node fails, its children must be visited by **one** of the mules to restore the lost data, which can be viewed as a mule assignment per node for the single node failure, or per unique node failure combination for the multi-failures case. In addition to T, we must find the location of all mules $\bar{m}$, and an assignment of each node $v \in V$ to a mule $m_i \in \bar{m}$ that minimizes the total travel cost of all mules. Formally, for $\beta > 1$, let $t(m_i, \delta(v, T))$ be the shortest path tour that includes mule m_i and the children of node v that mule m_i should visit. For $\alpha = 1$, the optimization problem is to find T and m such that $\sum_{v \in V} \sum_{m_i \in \bar{m}} |t(m_i, \delta(v, T))|$ is minimized.

We consider two network models, *complete* graphs and *unit disc* graphs. In the complete graph model, there is a directed edge between any pair of nodes in the graphs while in the unit disc graph model, there is an edge if and only if $d(u, v) \leq 1$, where $d(u, v)$ is the Euclidean distance between nodes u and v.

A summary of symbols used throughout this papers are depicted in Table 1.

m	The mule placement in T
$\delta(v, T)$	The children of node v in tree T.
$\lvert t(m, \delta(v, T))\rvert$	The cost of the shortest tour visiting the children of node v in tree T starting from node m.
$c(m, r)$	Total cost of the data gathering tree when mule is placed at node m and root is placed at node r. The notation is used for topologies for which the cost of the solution solely depends on m and r.
$\pi(i, m, r)$	Number of times node i is visited by the mule for a given m and r.
$c(T)$	The cost of a tree solution T when the placement of m and r is given in advance.

Table 1: Symbol table.

1.1. Our contribution

To the best of our knowledge, this is the first work exploring the mule approach for avoiding data loss due to communication failures. Our results are summarized in the following table:

Underlying graph	Problem	Topology	Approximation Ratio
Complete	$(1,1)$-Mule	Arbitrary	$1+1/c, c>1$
	$(\alpha,1)$-Mule		$\min(3, 1+s^*)$, $s^* = \min_{v\in V} \frac{\max d(v,u)}{\min d(v,w)}$
	$(1,\beta)$-Mule		2
UDG	$(1,1)$-Mule	Line	OPT
	$(\alpha,1)$-Mule	Line	OPT
	$(1,1)$-Mule	Random Line	4
	$(1,1)$-Mule	Grid	$1+(2+\sqrt{2})/\sqrt{n}$

Table 2: Summary of results.

1.2. Paper Outline

The paper is organized as follows. In the next Section we discuss the previous related work to our problem. We analyze different variations of the mule problem under the complete graph model and the Unit Disc Graph model in Sections 3 and 4, respectively. Section 5 outlines a possible distributed implementation of our algorithms. In Section 6 we present simulations of our algorithms under different network settings and conclude in Section 7.

2. Related Work

Exploiting mobile data carriers (mules) in ad-hoc networks has received significant attention recently. The subject of major interest in most works is using the mules to relay and collect messages in sparse network settings, where adjacent sensors are far from each other, in order to substantially reduce the cost of indeterminate sensors communication and data aggregation. For example, Wu et al. [22], investigate how to use the mule architecture to minimize data collection latency in wireless sensor networks. They reduce this problem to the well-known k-traveling salesperson with neighborhood and provide a constant approximation algorithm and two heuristic for it. In a related paper by Ciullo et al. [8], the collector is responsible for gathering data messages by choosing the optimal path that minimizes the total transmitted energy of all sensors subject to a maximum travel delay constraint. In their model, each sensor sends different amount of data. The authors also use the k-traveling salesperson with neighborhood problem in their solution technique and prove both analytically and through simulation that letting the mobile collector come closer to sensors with more data to transmit leads to significant reduction in energy consumption. Cheong et al. [6] investigate how to find a data collection path for a mobile base station moving along a fixed track in a wireless sensor network to minimize the latency of data collection. Levin et al. [17] considered the problem where the goal was to minimize the mules' traveling distance while minimizing the amount of information uncertainty caused by not visited a subset of nodes by the mule. A supplementary paper by Jea et al. [14] studies the practical advantages of offloading the collection using multiple data mules. A survey by Di Francesco et al. [9] covers the different aspects, methodologies and challenges for data collection in wireless sensor networks.

Another key aspect we discuss is using mules as backup mechanism for data loss resiliency in case of sensor failures. In [18], the authors propose a mechanism for backing up cell phone data using mobile sensor nodes. The goal of their protocol and infrastructure is to prevent losing data when the cell phone is lost, malfunction or stolen. Another approach for handling data loss in sensor networks is to construct a topology with path redundancy, where multiple paths connect each pair of nodes and serve as a backup mechanism in the case of node failure. In [15], Kim et al. propose a new algorithm based on results from algebraic graph theory, which can find the critical points in the network for single and multiple failure cases. They present simulations that examine the correlation between the number of critical points and sensor density. In [23], the authors proposed to build a biconnected communication graph where each pair of nodes in the network has at least two node disjoint paths between them, and thus, failure at any single node does not partition the network.

Multiple works in ad-hoc network examine the performance of graph related communication algorithms under linear or grid network topologies. The justification to explore such topologies is that multiple algorithms have been tested under realistic network conditions. In [11], Fraser et al. explore the usage of sensor networks for bridge monitoring. They build a continuous monitoring system, capable of handling a large number of sensor data channels and three video signals and deployed on a four-span, 90-m long, reinforced concrete highway bridge. In [13], Jawhar et al. consider a protocol for linearly structured wireless sensors to decrease installation, maintenance cost, and energy requirements, in addition to increasing reliability and improving communication efficiency. Their protocol takes advantage of the unique characteristics of linear networks and is well suited to be used in many sensor applications such as monitoring of international borders, roads, rivers, as well as oil, gas, and water pipeline infrastructures.

3. Complete graphs

In this section, we study the (α, β)-Mule problem under the complete graph model, where the underlying graph structure is complete (i.e., there is an edge between any pair of nodes) and the network topology is arbitrary.

3.1. $(1,1)$*-Mule problem in complete graphs*

Let S be a star over V and r be its root. We claim the following:

Lemma 1. *The optimal structure for the data gathering tree for the* $(1,1)$*-Mule problem on complete graphs is a star rooted in one of the nodes of* V*.*

Proof. For any data gathering tree each node in $V \setminus \{r\}$ must be traversed at least once. The proof follows since the travel distance of the mule for a star is:

$$|t(m, \{v\})| = \begin{cases} 0 & v \neq r \\ \text{Length of shortest tour} & \text{otherwise} \\ \text{over } V \setminus \{r\} & \end{cases}$$

is optimal. $\square$

Lemma 1 implies that the $(1,1)$-Mule problem is equivalent to the problem of finding a node $r \in V$ and a tour over $V \setminus r$, such that the cost of the tour is minimized. We use this fact to prove the $\mathcal{NP}$-completeness of the $(1,1)$-Mule problem. Consider the standard decision TSP problem:

Given a set S of n points, and length K, we need to find whether exist a cycle that goes through all points in S whose length is at most K? The decision version for the $(1,1)$-Mule problem is as follows: *given a set P of n points, and parameter L, we need to find whether we can remove one of the points so the cycle for the remained points will be of length at most L?*

Claim 2. *The $(1,1)$-Mule problem is $\mathcal{NP}$-complete.*

Proof. It is easy to see that the problem is in NP. We only show TSP $\leq_P(1,1)$-Mule . Given n points and parameter K from TSP instance, we construct the instance for our problem as follows. We set P to contain S and one more point x. The parameter L will be equal to K. We put point x far way from all other points of P so that the distance from x to any of them will be more than K. Clearly, there is a solution to $(1,1)$-Mule problem for P and L if and only if there is solution to TSP problem. □

Next, we present an approximation algorithm for the problem.

Lemma 3. *For any fixed $c > 1$, there is an $1+\frac{1}{c}$-approximation algorithm for $(1,1)$-Mule problem.*

Proof. Using the $1+\frac{1}{c}$-approximation algorithm for TSP [1], we can search for $r \in V$ that minimizes $|t(m,\delta(r))|$, where m is picked arbitrarily from $V \setminus \{r\}$. The running time is is $O(n(\log n)^{O(c)})$. □

We remark that alternative implementation can use Christofides's $\frac{3}{2}$-approximation algorithm [7] for finding the tour. The running time is $O(n^3)$.

3.2. $(\alpha,1)$-Mule problem in complete graphs

By similar argument as in Lemma 1, it is easy to see that the optimal topology for $(\alpha,1)$-Mule is a star rooted as some node r. We introduce Algorithm Build Tree 1. Let t_{opt} be the optimal

```
Build Tree 1
1 For each node $v \in V$, calculate $s(v) = \frac{\max_{u \in V \setminus \{v\}} d(v,u)}{\min_{w \in V \setminus \{v\}} d(v,w)}$, the ratio between the maximum to the minimum
  edge with respect to $v$. Set $r$ to be the node that minimizes this ratio and let $s^* = s(r)$ (ties are broken
  arbitrarily).
2 Set $T$ to be a star rooted at $r$.
3 Pick an arbitrary node $v \neq r$ and set $m = v$.
4 Find tour $C$ on $V \setminus \{r\}$ using the algorithm from [1].[1]
5 Emit $T, m, C$.
```

tour, r_{opt} be the root of the optimal tour, t be the tour produced by Algorithm Build Tree 1, and P_α be a permutation of α nodes.

Lemma 4. *Algorithm* Build Tree 1 *is a $(1+s^*)$-approximation algorithm for $(\alpha,1)$-Mule on complete graphs.*

Proof. We prove the claim by mapping, showing that for each combination of node failures P_α, either the mule travel costs of t_{opt} and t are the same, or that there exists a bijection from a permutation in t_{opt} to a permutation in t such that the solution's cost increases by at most $(1+s^*)$,

[1]This step in the algorithm can be accomplished by any approximation algorithm for TSP, e.g., [7].

where s^* is defined in Algorithm Build Tree 1. Let $V(P_\alpha)$ be the nodes that are traversed when the nodes in P_α fail. Clearly the solutions costs are the same if $r \notin P_\alpha$ and $r_{opt} \notin P_\alpha$ or $r \in P_\alpha$ and $r_{opt} \in P_\alpha$. For $r_{opt} \in P_\alpha$ and $r \notin P_\alpha$ the cost of t is 0 (since the tree has a form of star), while the cost of t_{opt} is the optimal tour over $V(P_\alpha)$; the opposite stands for $r_{opt} \notin P_\alpha$ and $r \in P_\alpha$. We show that for this case, for each combination P_α in t there is a combination $P_\alpha^{'}$ formed by adding twice (forward and back) the edge $e(r, r_{opt})$ to the solution that the new cost is at most $1 + s^*$ times the cost of t_{opt}. Clearly, each edge that connects r to the tour costs at least $\min_{u \in V \setminus \{r\}} d(r, u)$ and the new edge costs at most $\max_{u \in V \setminus \{r\}} d(r, u)$. Therefore, the cost of the new tour is at most $|t_{opt}| + 2\max_{u \in V \setminus \{r\}} d(r, u) = |t_{opt}| + 2s^* \min_{u \in V \setminus \{r\}} d(r, u) \leq |t_{opt}|(1 + s^*)$. Last equality holds since $|t_{opt}| \geq 2\min_{u \in V \setminus \{r\}} d(r, u)$. □

An alternative approach to this solution, is to select r that minimizes the length of minimum edge $e(r, w), \forall w \in V \setminus \{r\}$ with r as one of the endpoints. Similar analysis to the above yields $(1 + \frac{2s(r)}{n-\alpha})$-approximation ratio. This is because $t_{opt} \geq (n - \alpha)\min_{w \in V \setminus \{r\}} d(r, w)$ and the cost of new tour is $|t_{opt}| + 2\max_{u \in V \setminus \{r\}} d(r, u) = |t_{opt}| + 2s(r)\min_{w \in V \setminus \{r\}} d(r, w) \leq |t_{opt}| + 2s(r)\frac{|t_{opt}|}{n-\alpha} = (1 + \frac{2s(r)}{n-\alpha})|t_{opt}|$. Note that $s(r)$ does not necessary minimize maximum edge to minimum edge ratio.

Next, by carefully choosing r, we explain how to obtain a 3-factor approximation to our problem for a fixed value of α. Select r that maximizes the average cost of minimal edge (u, v) for each combination of $\alpha - 2$ failures. That is, per each node u and every edge (u, v) we calculate the number of times $t(u, v)$ (per each combination of $\alpha - 2$ failures) the edge (u, v) will be minimum edge from u. Next, we compute $ct(u) = \sum_{v \in V} d(u, v) \cdot t(u, v)$. Take r to be the node that maximizes $ct(r)$. If we consider the optimal solution OPT (which, according to the definition, contains many tours), then we notice that the sum of all edges' lengths that connect r in all tours is larger than $ct(r)$, since it must be equal at least the sum of all minimums. Thus, the total traveling distance in OPT is $|OPT| \geq 2ct(r)$. On the other side, by definition $ct(r_{opt}) \leq ct(r)$. The cost of new solution when adding r_{opt} is $|OPT| + 2c(r_{opt}) \leq |OPT| + 2ct(r) \leq 3|OPT|$.

3.3. $(1, \beta)$-Mule problem in complete graphs

In this section, we show how to solve the $(1, \beta)$-Mule problem on the complete Euclidean graph.

Lemma 5. *Algorithm* Build Tree 2 *produces is a 2-approximated solution for the $(1, \beta)$-Mule problem.*

Proof. Let C^i_{OPT} be the optimal tour traveled by mule m^i. By the construction of the algorithm and by the definition of minimum spanning tree: $\sum_i^\beta |T^i| \leq \sum_i^\beta |C^i_{OPT}| = OPT$. Let C^i be the tour constructed by traversing the nodes T^i using a depth-first-traversal. We have $\sum_i^\beta |C^i| \leq \sum_i^\beta 2|T^i| \leq 2OPT$. □

4. Unit disc graphs

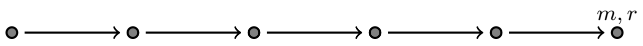
In this section, we study the different variation of the $(1, 1)$-Mule problem under the Unit Disc Graph model, where any two nodes $u, v \in V$, can communicate if and only if $d(u, v) \leq 1$.

Build Tree 2

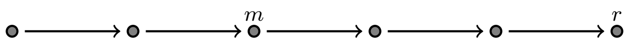
```
1 foreach $v \in V$ do
2     Find optimal spanning tree $T_v$ on $V \setminus \{v\}$
3     Let $T_v^1, T_v^2, \dots, T_v^\beta$ be the set of trees created by removing the $\beta - 1$ heaviest edges from $T_v$
4     Assign the nodes in $T_v^i$ to mule $m_i$.
5 end
6 Let $v$ be the node that minimizes $\sum_{i=1}^{\beta} |T_v^i|$.
7 Set $T$ to be a star rooted at $v$.
8 Emit $T, \bar{m} = \{m_v^1, \dots, m_v^\beta\}$.
```

4.1. $(1,1)$-*Mule problem in line topology*

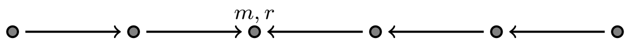
Here, n nodes, with unit distance between them, are placed in the Euclidean plane. The construction ensures that a node can communicate only with its adjacent neighbors. For the line topology under those communication constraints, only the placement of the root r is required to define the structure and orientation of the tree. Thus, the cost of a solution is solely determined by the placement of r and m. For clarity, we number the nodes from 1 to n and use m and r as the indices of the mule and the root in the solution. From symmetry, we assume $r \geq m$, since $c(m,r) = c(n-m+1, n-r+1)$, where $c(m,r)$ is the cost of the optimal solution when the mule is located at m and the root is placed at r when the topology is entirely determined by the location of r (e.g., line). A sample instance of the problem is depicted in Figure 2.



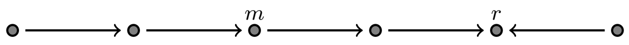
(a) $c(m,r) = 0 + 10 + 8 + 6 + 4 + 2 = 30$



(b) $c(m,r) = 0 + 4 + 2 + 0 + 2 + 4 = 12$



(c) $c(m,r) = 0 + 4 + 4 + 4 + 6 + 0 = 18$



(d) OPT. $c(m,r) = 0 + 4 + 2 + 0 + 6 = 12$.

Figure 2: Line topology illustration. The figure contains 6 nodes with two different solutions for T and two different choices for locating the mule. Each value in sum represents the cost of failing node i, $i \in \{1, ...6\}$.

Lemma 6. *For the line topology, the optimal placement for the root r is $n-1$.*

Proof. Assume $m < r$, if a node $i \in V$ fails, we have four cases:

1. $i < m$, the cost is $2(m - i + 1)$ regardless to the location of r.
2. $i < m < r$, the cost is $2(r - m - 1)$.
3. $r < i, i \neq n$, the cost is $2(r - m + 1)$.
4. $r < i, i = n$, the cost is 0 (since i has no children).

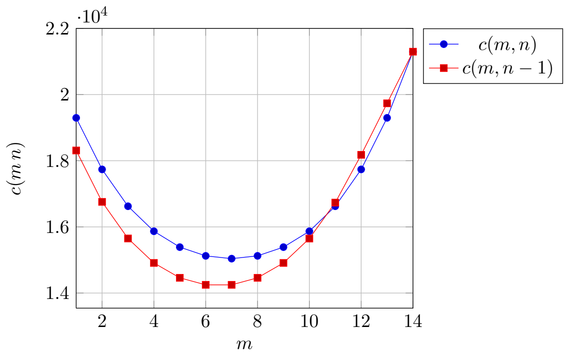
The claim follows since we want to minimize the number of nodes that are placed after r, but can use the fact that the cost is zero for $r < i = n$. ☐

Lemma 7. *For line topology, the optimal placement for the mule is* $\lceil \frac{n}{2} \rceil$.

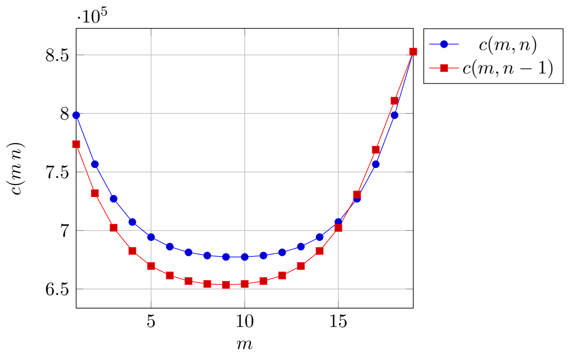
Proof. For optimality $r = n - 1$. Then $c(m, r) = 2(\sum_{i=1}^{m-1} i + \sum_{i=1}^{n-m-1} i + 2)$ is maximized for $\lceil \frac{n}{2} \rceil$. ☐

4.2. $(\alpha, 1)$-Mule in the line topology

In this section, we show how to handle α simultaneous failures on the line topology. We show a formula for calculating $c(m, r)$ and prove that the values that minimize $c(m, r)$ are $m = \frac{n}{2}$ and $r = n-1$. The highlights of the proof are as follow: we show that for $r = n$, $c(m, n)$ is monotonically decreasing for $m < \frac{n}{2}$ and monotonically increasing for $m > \frac{n}{2}$, which implies a global minimum for $m = \frac{n}{2}$. Next, we extend the proof and show that this global minimum for $r = n - 1$ is still $m = \frac{n}{2}$. To illustrate the concepts behind the proof, the costs of $c(m, n)$ and $c(m, n - 1)$ for varying values of m are given in Figure 3.



(a) $c(m, n)$ for $n = 14$.



(b) $c(m, n)$ for $n = 20$.

Figure 3: $c(m, r)$ for varying values of m.

First, we introduce some basic definitions. We define a *direct* visit when the mule visits node i where i is the leftmost node if $i < m$ or the rightmost node if $i > m$. Let $\pi(i, m, r)$ be the number of times the mule at placement m directly visits node i for root placement r. We separate between left and right movement and define $\pi_l(m, r) = \sum_{i=1}^{m-1} \pi(i, m, r)$ and $\pi_r(m, r) = \sum_{i=m+1}^{n} \pi(i, m, r)$ to be the number of times that the mule must travel left or right when placed at location $m \in [1, n]$.

We begin by showing an optimal but inefficient algorithm for the problem:

Lemma 8. *For $m \in [1, n-2]$, $c(m, n-1)$ has a closed formula, which can be calculated in polynomial time.*

Proof. First note that we only visit node at i, when node at $i + 1$ fail. For $m < i < n - 2$ we have $\pi(i, m, n-1) = \sum_{j=1}^{n-i-2} \binom{i-1}{\alpha-j} + \sum_{j=1}^{n-i-1} \binom{i-1}{\alpha-j-1}$. The left expression represents the case where node at placement n did not fail and the right expression represents the case where node at placement n did fail. For $i = n-2$ we have $\pi(n-2, m, n-1) = \binom{n-3}{\alpha-2} + \binom{n-3}{\alpha-1}$. For $i = n$: $\pi(n, m, n-1) = \binom{n-2}{\alpha-1}$. The expression $\pi(i, m, n-1)$ for $i < m$ represents the case where j consecutive nodes from the right side of i failed and equals $\sum_{j=1}^{\min(\alpha-1,i-1)} \binom{n-(i+1)}{j}$. Let $d(m,i)$ be the Euclidean distance between m and i, the cost is $c(m,n-1) = \sum_{i=1}^{n} \pi(i,m,n-1) \cdot d(m,i)$. which we can calculate in polynomial time. □

From Lemma 6 we know that the optimal placement for the root is $n-1$. Therefore, to find the optimal solution, we can search for the value of m that minimizes $c(m,n-1)$. Using dynamic programming and the memoization table, in $O(n^2)$ time we can compute the values of $c(i,j)$, and calculate the total cost. Thus, the running time of the algorithm is $O(n^2)$.

Now we show that the optimal cost is obtained for $m = \frac{n}{2}$ and $r = n-1$. First we claim the following:

Lemma 9. *For $m < i$, $\pi(i,m,r) = \pi(i,m+1,r)$ and for $m > i$, $\pi(i,m,r) = \pi(i,m-1,r)$.*

Proof. As long as $m \neq i$ the orientation of the mule with respect to i does not change. □

Lemma 10. *$c(m+1,r) = c(m,r) + \pi_l(m,r) + \pi(m,m+1,r) - \pi_r(m,r)$ and $c(m-1,r) = c(m,r) - \pi_l(m,r) + \pi(m,m-1,r) + \pi_r(m,r)$.*

Proof. Let $d(m,i)$ be the distance between m to i. Thus, $c(m,r) = \sum_{i=1}^{m} \pi(i,m,r)d(m,i) + \sum_{i=m+1}^{n} \pi(i,m,r)d(m,i)$. Assume we place the mule at location $m+1$. From Lemma 9 we have $c(m+1,r) = \sum_{i=1}^{m-1} \pi(i,m,r)(d(m,i)+1) + \pi(m,m+1,r) + \sum_{i=m+1}^{n} \pi(i,m,r)(d(m,i)-1)$. Since $d(m,m)$=0 we get $c(m+1,r) - c(m,r) = \pi_l(m,r) + \pi(m,m+1,r) - \pi_r(m,r)$. And when placing the mule in $m-1$ we obtain $c(m-1,r) - c(m,r) = -\pi_l(m,r) + \pi(m,m-1,r) + \pi_r(m,r)$. and the claim follows. □

Next, we show that:

Lemma 11. *For $r = n$, $\pi_l(\frac{n}{2}, n) = \pi_r(\frac{n}{2}, n)$.*

Proof. When $r = n$, we show that for each node on the right side r, there is a bijection to a node on the left side l, such that $\pi(r,m,n) = \pi(j,m,l)$. This means that the number of times the mule travels specifically to l is equal to the number of times it travels to r (note that this does not necessarily imply that the distances of l and i from m are the same or that they equally contribute to $c(m,r)$). To see this, we look at the number of permutations when some node $r > \frac{n}{2}$ fails. We travel directly to r when a set of j consecutive nodes with respect to r fail (i.e., $r+1, r+2, \ldots, \min r+j, \alpha)$ and $\alpha - j$ nodes that are on the left hand side of r fail. Formally, this is equal to $\pi(r,m) = \sum_{j=1}^{n-r} \binom{r-1}{\alpha-j}$. For some node $l < \frac{n}{2}$, we travel to l when a set of j consecutive nodes from the leftmost node fail (i.e., $1, 2, \ldots, \min j, \alpha$), and another j node that are on the right hand side of l fail. Formally, this is equal to $\pi(l,m) = \sum_{j=1}^{l-1} \binom{n-(l+1)}{\alpha-j}$. We have the expressions equal for $l = n-i+1$ and the claim follows. □

Lemma 12. *For increasing m $\pi_l(m,r)$ is monotonically increasing and $\pi_r(m,r)$ is monotonically decreasing.*

Proof. Regardless of the mule placement, from Lemma 9 and as long as $i > m$, the number of times the mule travel to a specific node is constant. Since increasing m means less nodes are on the right hand side, with no change in orientation with respect to m, $\pi_l(m,r)$ is increasing. Since more nodes are added from the left side of m, $\pi_r(m,r)$ is decreasing. □

Lemma 13. *For $r = n$, the function $c(m, n)$ has global minimum at $\lceil \frac{n}{2} \rceil$.*

Proof. Follows from Lemmas 11 and 12. $\square$

We have shown that for $c(m, n)$ yields optimal value for $m = \frac{n}{2}$. To complete the proof, we turn to handle the case of $r = n - 1$.

Lemma 14. *For $r = n - 1$, the function $c(m, n - 1)$ has global minimum at $\lceil \frac{n}{2} \rceil$.*

Proof. For $l < m$, $\pi(m, l)$ is not impacted by this change. However, for each node $r < n - 2$ on the right of m, we separate to two cases: directly visiting r when node n fails or nodes n and $n-1$ do not fail. Formally, $\pi(r, m, n-1) = \sum_{j=1}^{n-r-2} \binom{r-1}{\alpha-j} + \sum_{j=1}^{n-r-1} \binom{r-1}{\alpha-j-1} = \sum_{j=1}^{n-r-2} \binom{r-1}{\alpha-j} + \sum_{j=2}^{n-r} \binom{r-1}{\alpha-j} = \sum_{j=1}^{n-r-2} \binom{r-1}{\alpha-j} + \sum_{j=n-r-1}^{n-r} \binom{r-1}{\alpha-j} + \sum_{j=2}^{n-r-2} \binom{r-1}{\alpha-j} = \sum_{j=1}^{n-r} \binom{r-1}{\alpha-j} + \sum_{j=2}^{n-r-2} \binom{r-1}{\alpha-j}$. For $r = n - 2$, we have $\pi(n-2, m, n-1) = \binom{n-3}{\alpha-2}$. Finally, for $r = n$, $\pi(m, m, n-1) = \binom{n-2}{\alpha-1}$. Thus, we obtain $\pi_r(m, n) - \pi_r(m, n-1) = \binom{n-3}{\alpha-1} - \sum_{r=m+1}^{n-3} \sum_{j=2}^{n-r-2} \binom{r-1}{\alpha-j} = \Delta$. To complete this proof, all we have to show is that the function $c(m, n-1)$ is monotonicity increasing when $m > \frac{n}{2}$ and monotonicity decreasing when $m < \frac{n}{2}$, which means that the minimum is achieved at $m = \frac{n}{2}$.

Combining Lemmas 10 and 11, we have to show that: $0 \leq c(m+1, n-1) - c(m, n-1) = \pi_l(m, n-1) - \pi_r(m, n-1) + \pi(m+1, m, n-1) = \pi_l(m, n-1) - (\pi_r(m, n) - \Delta) + \pi(m+1, m, n-1) = \pi(m+1, m, n-1) + \Delta$ and that: $0 \leq c(m-1, n-1) - c(m, n-1) = -\pi_l(m, n-1) + \pi_r(m, n-1) + \pi(m-1, m, n-1) = \pi(m-1, m, n-1) - \Delta$.

Clearly the first expression is true since $\pi(m+1, m, n-1) + \Delta$ is positive. To complete the proof, we show that $\Delta \leq \pi(m-1, m, n-1)$. Reversing the order of summation yields $\Delta = \binom{n-3}{\alpha-1} - \sum_{j=2}^{n-m-3} \sum_{r=m+1}^{n-3-(j-1)} \binom{r-1}{\alpha-j}$. Using the binomial coefficient identity: $\sum_{i=0}^{n} \binom{i}{c} = \binom{n+1}{c+1}$ we get $\Delta = \binom{n-3}{\alpha-1} - \sum_{j=2}^{n-m-3} (\binom{n-3-(j-1)}{\alpha-j+1} - \binom{m}{\alpha-j+1}) = \binom{n-3}{\alpha-1} - \sum_{j=0}^{n-m-5} (\binom{n-4-j}{\alpha-j-1} - \binom{m-2}{\alpha-j-1})$. Using the binomial coefficient identity $\sum_{i=0}^{c} \binom{n-i}{c-i} = \binom{n+1}{c}$ and assuming $n - m - 5 \geq \alpha$ we obtain $\Delta = \binom{n-3}{\alpha-1} - \binom{n-3}{\alpha-1} + \sum_{j=0}^{n-m-5} \binom{m-2}{\alpha-1-j}$. Setting $m = \frac{n}{2}$, we have $\Delta = \sum_{j=0}^{\frac{n}{2}-5} \binom{\frac{n}{2}-2}{\alpha-1-j}$. Finally, by setting $m = \frac{n}{2}$ in $\pi(m-1, m, n-1)$ it results in: $\pi(m-1, m, n-1) = \pi(\frac{n}{2} - 1, \frac{n}{2}, n-1) = \sum_{j=1}^{\frac{n}{2}-1} \binom{n-(\frac{n}{2}-1+1)}{\alpha-j} = \sum_{j=0}^{\frac{n}{2}} \binom{\frac{n}{2}}{\alpha-1-j} \geq \Delta$ and the proof is complete. $\square$

We conclude with the following:

Theorem 15. *The optimal placement for $(\alpha, 1)$-Mule on the line topology is $r = n - 1$ and $m = \frac{n}{2}$.*

4.3. $(1, 1)$*-Mule problem in the random line topology*

In this section, we solve the $(1, 1)$-Mule problem on the random line, where n nodes are placed on a line with length $n \gg L$ such that the distances between adjacent nodes are sampled from a predefined distribution function, i.e., the maximum distance is 1. The communication model is Unit Disc Graph, which means that an edge is formed between two nodes u, v if and only if $d(u, v) \leq 1$. Note that this implies that the graph is connected. In what follows, we use the simplified assumption that the mule m and root r are positioned in the leftmost node of the line and that $L \in \mathbb{N}$.

Let T be the tree produced by Algorithm Build Tree 3, T_{opt} be the optimal tree and $c(T)$ and $c(T_{opt})$ be their costs, respectively. We define T_L as the tree over exactly L nodes such that the distance between adjacent nodes is exactly one; let $c(T_L)$ be its cost. Observe that in the algorithm, the set B represents the "backbone" nodes in T that are not leaves. We claim:

Algorithm 3: BUILD TREE 3

```
1  $V' = B = C = \{r\}$
2  $E' = \emptyset$
3  while $|C| \neq n$ do
4      Let $C$ be all nodes reachable by nodes in $B$.
5      Find furthest node $v$ that is reachable by nodes in $B$ .
6      Find node $u \in B$ that minimizes $d(u, v)$.
7      Add $v$ to $B$.
8      Add the edge $e(v, u)$ to $E^{'}$.
9      For each $w \in C \setminus \{v\}$, add a directed edge $e(w, v)$ to $E^{'}$.
10     $V' = V' \cup C \cup \{v\}$.
11 end
12 Emit $T = (V', E')$.
```

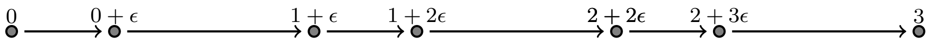


Figure 4: Placement where approximately $2L$ nodes are required to cover an area with length L.

Lemma 16. $c(T_L) \leq c(T_{opt})$.

Proof. Note that at least L nodes are required to cover an area of length L and that each unit interval on the line must contain at least one node. Therefore, we can convert any tree to T_L by mapping one of the nodes in interval $[i, i + 1]$ to the node at location i in T, and drop all other nodes in that interval. Since $m = 0$, this conversion reduces the overall cost of the solution. This implies, a fortiori, that $c(T_L) \leq c(T_{opt})$. □

Lemma 17. $|V(T)| \leq 2L$.

Proof. Let v and l be two non-leaf nodes that are selected in two consecutive iterations of Algorithm BUILD TREE 3, and v_x and l_x be their x coordinates on the line, respectively. The algorithm will converge in most slowest rate when l_x is closest as possible to v_x, but since l is the furthest node in the range $[v_x, v_x + 1]$ it means the non-leaf node that will be selected after l must be in $[v_x + 1, v_x + 1 + \epsilon]$. Thus, in the worst case, the algorithm covers a unit distance in two iterations, which means that it completes after at most $2L$ steps. See the illustration in Figure 4. □

Lemma 18. $c(T) \leq 4c(T_L)$.

Proof. By definition $c(T_L) = 2\sum_{i=1}^{L} i = L(L+1)$. Let i_x be the coordinate of non-leaf node selected in iteration i in Algorithm BUILD TREE 3, we have: $c(T) \leq 2\sum_{i=1}^{2L} i_x \leq 2\sum_{i=1}^{2L} i = 2L(2L + 1)$. The last inequality follows since we stretch a line of length L to a line of length $2L$. □

Therefore, we have:

Lemma 19. *Algorithm* BUILD TREE 3 *yields a* 4*-approximation for the* $(1, 1)$*-Mule problem.*

4.4. $(1,1)$-*Mule problem in grid topology*

Next, we assume that the nodes of the graph are deployed on a $\sqrt{n}\times\sqrt{n}$ grid and have unit transmission radius.

Let d_v be the degree of node $v\in V$ and $d_{\max}$ be the maximum degree of any node in the input graph G and $v_{i,j}$ be the location of node at coordinates i,j, we claim:

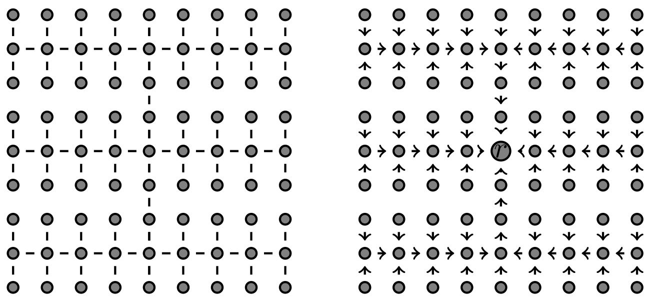
Lemma 20. *For a specific mule placement m, the approximation ratio of any tree to the* $(1,1)$*-Mule problem is at most* $d_{\max}$.

Proof. Clearly, for any algorithm all non root nodes must be visited by the mule. In the worst case, that incurs the least value is when a node v has a single child in T. Then, the mule's tour only covers one node. In the best case, each tour includes all children of v in G, which is obliviously bounded by its degree. The claim follows since the ratio between the cost node v incurs in the worst solution and the optimal solution is at most d_v and since all children of v must be visited by the mule in the algorithm. □

Next, we show a lower bound on OPT.

Lemma 21. $OPT \geq 2\frac{\sum_{i=1}^{\sqrt{n}}\sum_{j=1}^{\sqrt{n}} d(v_{i,j},v_{\frac{\sqrt{n}}{2},\frac{\sqrt{n}}{2}})}{3}$.

Proof. Let $m_{x,y}$ be the location of the root, and assume that we use a zig-zag tree as a solution (see Figure 6). Clearly, the cost is $2\sum_{i=1}^{\sqrt{n}}\sum_{j=1}^{\sqrt{n}} d(v_{i,j},m_{x,y})$, which is optimized by $x=\frac{\sqrt{n}}{2}$ and $y=\frac{\sqrt{n}}{2}$. The proof follows by combining the fact that except from the root, for any tree in the grid $d_{max}=3$, and from Lemma 20. □



(a) Stars step (b) Orientation step

Figure 5: Illustration of Algorithm Build Tree 4.

Next, we present Algorithm Build Tree 4 that constructs a tree with almost optimal cost. To maximize the number of nodes visited per failure we try to produce a tree with maximum number of leaves. We use the principals presented at [3] and build the tree on the top of multiple consecutive stars. Let c be the cost of Algorithm Build Tree 4 and s be the cost of the zig-zag tree. We

```
Build Tree 4
1 Stars Build adjusted stars for all nodes with coordinates $(x,y)$ such that $1\equiv y \bmod 3$ (Figure 5a).
2 Orientation Connect stars with grid orientation (Figure 5b).
```

show:

Lemma 22. *Algorithm* Build Tree 4 *is a* $1+\frac{2+\sqrt{2}}{\sqrt{n}}$*-approximation algorithm.*

Proof. On the one side $c = 2\sum_{i=1}^{\sqrt{n}}\sum_{j=1|1\equiv j \bmod 3}^{\sqrt{n}}(d(v_{i,j},m) + (1+\sqrt{2})) + 2\sum_{j=1|1\equiv j \bmod 3}^{\sqrt{n}} j \leq 2\frac{\sum_{i=1}^{\sqrt{n}}\sum_{j}^{\sqrt{n}} d(v_{i,j},m)}{3} + 2\sum_{i=1}^{\sqrt{n}}\sum_{j=1|1\equiv j \bmod 3}^{\sqrt{n}}(1+\sqrt{2}) + \frac{2}{3}n = OPT + \frac{2}{3}n(2+\sqrt{2})$. On the other side, since we can project all nodes in the zig-zag tree solution to the x-plane and place m at $(\frac{\sqrt{n}}{2}, 0)$ we have $s = 2\sum_{i=1}^{n}(i - \frac{\sqrt{n}}{2}) \geq n^2 - n\sqrt{n} \geq 2n\sqrt{n}$. The last inequality holds for $n > 9$. Since the projection reduces the travel cost of the solution, together with Lemma 20 we have $OPT \geq \frac{s}{3} \geq \frac{2n\sqrt{n}}{3}$ Hence, $c \leq (1+\frac{2+\sqrt{2}}{\sqrt{n}})OPT$. □

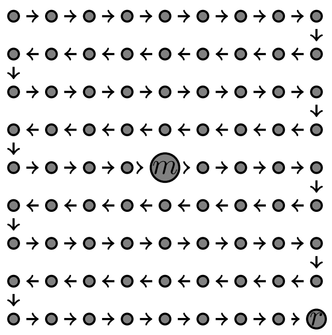


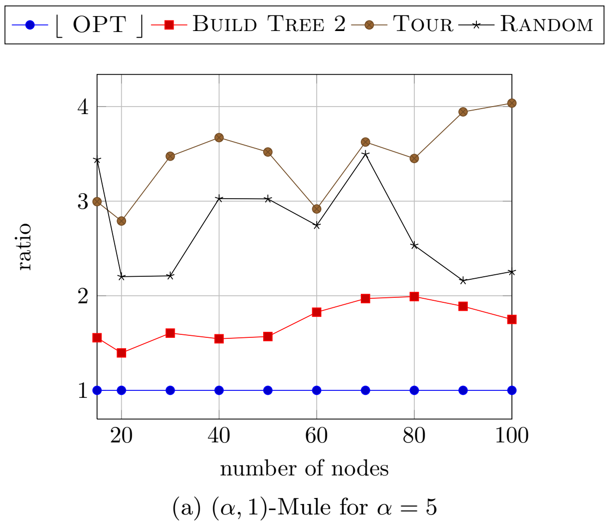
Figure 6: Zig-zag tree with cost $2\sum_{i=1}^{\sqrt{n}}\sum_{j=1}^{\sqrt{n}} d(v_{i,j},m)$.

5. Distributed Implementation

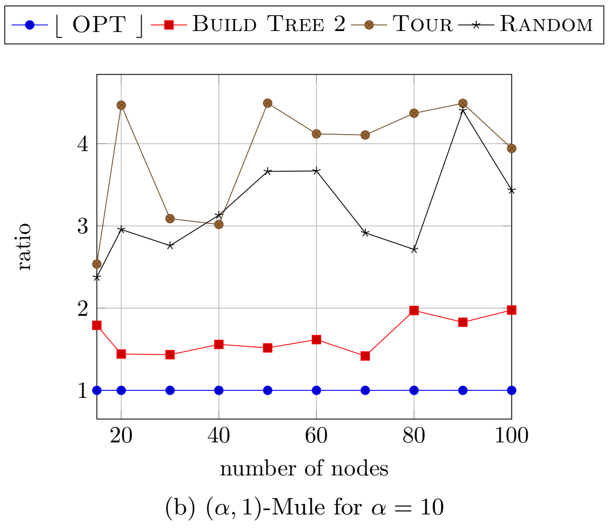
In order to make our solutions feasible, i.e. to allow them to work in real life node deployments, we outline how it is possible to implement them in a decentralized (distributed) (without the need for coordination by a central unit) and local, based on neighbor knowledge manner. In the proposed distributed implementations we make a use of the work [2]. The paper [2] shows how to find a leader in a distributed fashion (and also minimum spanning tree) in a network with n nodes in $O(n)$ time using $O(n \log n)$ messages. To establish connectivity, can follow two different approaches as described in [19]. The first, described in Dolev et al. [10] forms a temporary underlying topology in $O(n)$ time using $O(n^3)$ message. The second (better) approach is given by Halldórsson and Mitra [12] that shows how to do this in $O(poly(\log\gamma, \log n))$, where γ is the ratio between the longest and shortest distances among nodes. After the topology is established, we can use leader-election algorithm by Awerbuch [2] that can compute all other relevant information in the network, i.e. choose an appropriate root r or find the tour. Given each sensor knows the total number of nodes in the network, the distributed implementation of Build Tree 4 algorithm only requires the local GPS coordinates of each sensor. To retrieve this information, we can apply Peleg et al. [20] distributed algorithm for finding the graph's diameter and propagate it to all sensors.

6. Simulations

In what follows, we describe the simulation results of the various algorithms and network models proposed in this paper. We have implemented all algorithms described throughout the paper using standard simulation software written in `C#` and conducted multiple experiments on different topologies. For each experiment, we have calculated the mean solution cost after conducting 50



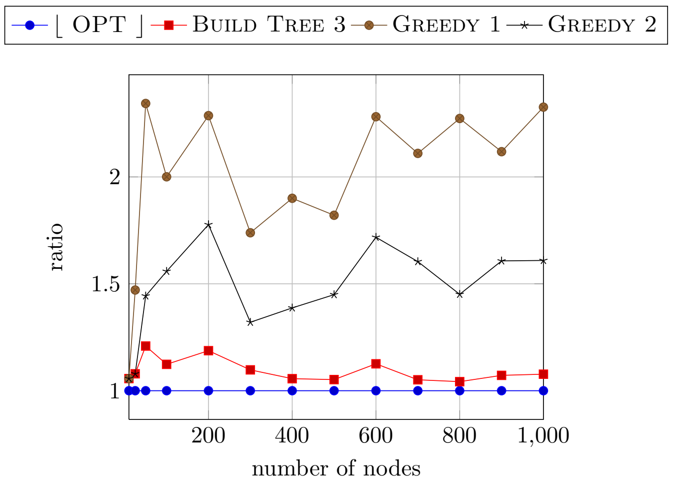
(a) $(\alpha, 1)$-Mule for $\alpha = 5$



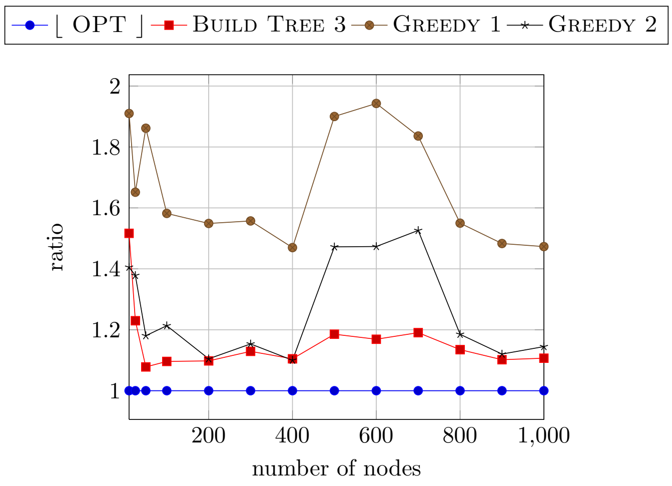
(b) $(\alpha, 1)$-Mule for $\alpha = 10$

Figure 7: Comparison between the cost of Algorithm BUILD TREE 2 and the cost of alternative efficient tree construction solutions.

iterations. For large networks, for which calculating the shortest TSP is not computationally feasible, we have used a TSP heuristic framework based on a genetic algorithm [16]. To show the clear advantages of using this paper algorithms we introduce the notation of lower bound OPT, $\lfloor \text{OPT} \rfloor$, which is calculated based on the different bounds we provide under the different network settings. In all simulations we compare the ratio of the proposed algorithm to the lower bound on OPT. In the first simulation (Figure 7), we investigated the variance of initiating different input trees in step 1 of Algorithm BUILD TREE 2. To produce the mule's tours we used the heuristic genetic algorithm from [4]. We compared our results to the following variations: TOUR, finding the optimal approximated tour over $n-1$ nodes using the heuristic algorithm from [4] and then taking the minimum spanning tree over those nodes, RANDOM, building a random tree, and $\lfloor \text{OPT} \rfloor$, using the minimal spanning tree instead of tours (thereby making its cost a lower bound on OPT). We provide results for 5 and 10 sensor failures, correspondingly. From the simulations, we can see that the rival algorithms substantially suffer from the increase in failures, which means higher cost of traveling tours with respect to Algorithm BUILD TREE 2. The results show that the bound proved in Lemma 5 holds and that in practice, might be even better. In the second simulation (Figure 8), we explored how different leader selection in step 4 of Algorithm BUILD TREE 3 impacts the total cost of the algorithm. We compared the results of our algorithm against three competitive algorithms: GREEDY1, randomly selecting one of the nodes as leader, GREEDY2, selecting the node closest to the rightmost leader and $\lfloor \text{OPT} \rfloor$, changing the distances between the adjacent nodes to L/n. In the simulations, we tested how the distribution function of the adjacent distance between nodes impacts the performance of the algorithm. In Figure 8a, we used the exponential distribution with mean 0.1 and in Figure 8b, the uniform distribution with mean 0.5. Reviewing the experiment data, we noticed that the burstness of the exponential distribution causes increases the travel distance of the mule, thereby increasing the overall cost of the solutions. Finally, note that the actual approximation ratio of Algorithm BUILD TREE 3 is much lower than the one proved



(a) Results for exponential distribution with mean 0.1.



(b) Results for uniform distribution with mean 0.5.

Figure 8: Ratio between the cost of algorithm Build Tree 3 against competitive algorithms

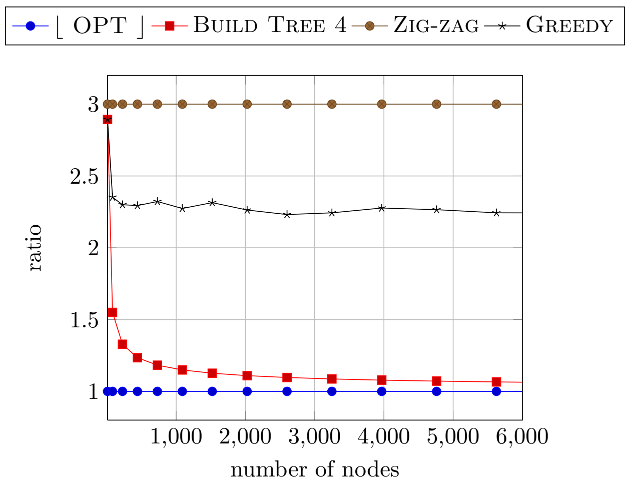
in Lemma 18, which may indicate that we can theoretically tighten the approximation ratio. In the final simulation (Figure 9), we compared the results of Algorithm Build Tree 4 against the following competitive algorithms: Zig-zag, using the zig-zag tree (see Figure 6), Greedy, using the minimum spanning tree and $\lfloor \mathrm{OPT} \rfloor$, using the zig-zag tree but diving the cost by 3 (see Lemma 20). We study two variations, placing the mule at the leftmost corner coordinate and placing the mule at the center. Its interesting to note that although the ratio between algorithms in both simulations remains the same, the actual cost was much higher when placing the mule at the corner.

7. Conclusions and Future Work

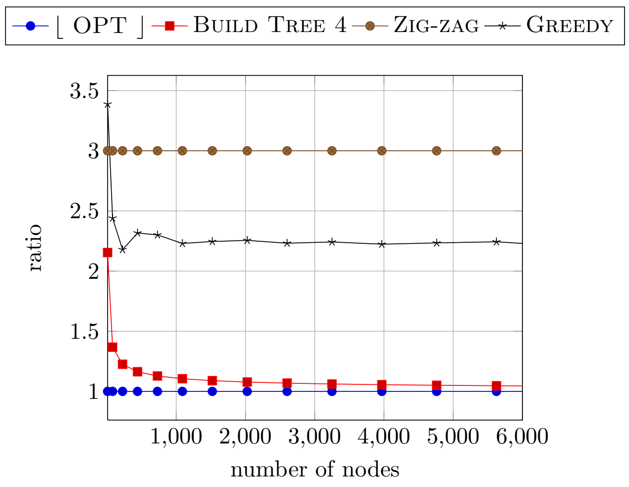
This work address the topological design of data mules usage for improving resiliency to data loss caused by network disasters. Our solutions involve constructing the optimal data gathering tree and finding the optimal node placement under multiple network structures, such as general graph, linear and grids. We use the topology characteristic to produce multiple approximation algorithms and validate their performance using simulations.

This paper emphasizes on the problem of minimizing the sum of distances the mule travels. Instead, we can explore minimizing the maximum traveling distance or time of the mule. Formally, we can define the $(1,1)$-Mule problem as follows: $\min_{T,m} \max_{v \in V} |t(m, \delta(v,T))|$. Interestingly, the given objective completely changes the complexity and algorithms of the problem. For example, while the optimal topology in the min-sum version was a star, in the min-max version it is a line that traverse all the nodes in the graph. In addition, we can find the optimal solution by carefully selecting the location of the mule, which means the problem is not NP-hard. It will be interesting to further explore this objective under different network criteria and to compare the solutions to the ones proposed in this paper.

Although we study the problem under varying network structures, we did not measured the impact of geographical surrounding or diverse hardware on the sensor durability. In the future,



(a) Results when mule is placed at center coordinates.



(b) Results when mule is placed at corner coordinates.

Figure 9: Ratio between the cost of algorithm BUILD TREE 4 against competitive algorithms.

we intend to add varying sensor resistances to our model by applying different failure probability function per sensor, which can help in modeling uneven and rough geographic conditions. Another interesting variation can explore the impact of transmission radius on the mule tour. That is, given some minimal transmission radius for the sensor, instead of visiting the actual sensor placement, the mule only visits the sensor surrounding. This work can reuse existing results [22, 8] to extend the algorithms proposed here.

Acknowledgement. This research was supported by the Engineering and Physical Sciences Research Council (EPSRC), United Kingdom and by the Israel Science Foundation (grant No. 317/15).

References

[1] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. In *Journal of the ACM*, pages 2–11, 1996.

[2] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *ACM STOC 1987*, pages 230–240, 1987.

[3] Randeep Bhatia, Samir Khuller, Robert Pless, and Yoram J. Sussmann. The full degree spanning tree problem. In *SODA*, pages 864–865, 1999.

[4] Kylie Bryant. *Genetic algorithms and the traveling salesman problem*. PhD thesis, 2000.

[5] Guner D. Celik and Eytan Modiano. Dynamic vehicle routing for data gathering in wireless networks. In *CDC*, 2010.

[6] Otfried Cheong, Radwa El Shawi, and Joachim Gudmundsson. A fast algorithm for data collection along a fixed track. In *COCOON*, pages 77–88, 2013.

[7] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Carnegie Mellon University, 1976.

[8] Delia Ciullo, Guner D. Celik, and Eytan Modiano. Minimizing transmission energy in sensor networks via trajectory control. In *WiOpt*, pages 132–141, 2010.

[9] Mario Di Francesco, Sajal K. Das, and Giuseppe Anastasi. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Trans. Sen. Netw.*, 8(1):7:1–7:31, August 2011.

[10] S. Dolev, M. Segal, and H. Shpungin. Bounded-hop energy-efficient liveness of flocking swarms. *IEEE Transactions on Mobile Computing*, 12(3):516–528, 2013.

[11] Michael Fraser, Ahmed Elgamal, Xianfei He, and Joel P Conte. Sensor network for structural health monitoring of a highway bridge. *J. of Comp. in Civil Engineering*, 24(1):11–24, 2009.

[12] Magnús M. Halldórsson and Pradipta Mitra. Distributed connectivity of wireless networks. In *PODC*, pages 205–214, 2012.

[13] Imad Jawhar, Nader Mohamed, Khaled Shuaib, and Nader Kesserwan. Monitoring linear infrastructures using wireless sensor networks. In *Wireless Sensor and Actor Networks II*, volume 264, pages 185–196. 2008.

[14] David Jea, Arun Somasundara, and Mani Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Distributed Computing in Sensor Systems*, volume 3560 of *LNCS*, pages 244–257. 2005.

[15] Tae-Hoon Kim, David Tipper, Prashant Krishnamurthy, and A Lee Swindlehurst. Improving the topological resilience of mobile ad hoc networks. In *DRCN*, pages 191–197, 2009.

[16] Michael LaLena. *Traveling Salesman Problem Using Genetic Algorithms.* PhD thesis.

[17] Liron Levin, Alon Efrat, and Michael Segal. Collecting data in ad-hoc networks with reduced uncertainty. *Ad Hoc Networks*, 17:71–81, 2014.

[18] Damien Martin-Guillerez, Michel Banâtre, and Paul Couderc. Increasing Data Resilience of Mobile Devices with a Collaborative Backup Service. Rapport de recherche, 2006.

[19] V. Milyeykovsky, Michael Segal, and Hanan Shpungin. Location, location, location: Using central nodes for efficient data collection in wsns. In *IEEE WIOPT*, 2013.

[20] David Peleg, Liam Roditty, and Elad Tal. Distributed algorithms for network diameter and girth. In *Automata, Languages, and Programming*, volume 7392 of *Lecture Notes in Computer Science*, pages 660–672. Springer Berlin Heidelberg, 2012.

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

[22] Weili Wu, Alade O. Tokuta, Wei Wang, Baraki H. Abay, R.N. Uma, and Donghyun Kim. Minimum latency multiple data muletrajectory planning in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 13(4):838–851, 2014.

[23] Zhongjiang Yan, Yilin Chang, Hai Jiang, and Zhong Shen. Fault-tolerance in wireless ad hoc networks: bi-connectivity through movement of removable nodes. *Wireless Communications and Mobile Computing*, 13(12):1095–1110, 2013.