Data Gathering in Faulty Sensor Networks Using a Mule

Stav Ashur[1]

1 **Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel**
stavshe@post.bgu.ac.il

Abstract

We study the problem mentioned in the title, assuming the underlying sensor network is a unit disk graph. That is, let S be a set of n sensors with transmission range 1. We wish to find a data gathering tree (i.e., a rooted spanning tree) for the network, and to augment it with a data mule based at one of the nodes of the tree. The mule's job is to collect the data from the children of a node u, if u is faulty. The goal is to find a gathering tree and to locate the mule at one of its nodes, so that the expected length of the mule's tour is minimized, where the mule can move freely in the plane. We present an $O(n \log n)$-time constant-factor approximation algorithm for this problem. Our algorithm is faster by a linear factor than the previous one due to Yedidsion et al. [5].

1 Introduction

Given a set S of n sensors with wireless capabilities deployed in the plane, one would like to gather the data collected by the sensors using a *data gathering tree* — a hierarchical structure that determines the paths in which data flows from the sensors to the storage center (i.e., a directed rooted tree). The transmission range of the sensors is 1, so our starting point is the *unit disk graph* (UDG) G induced by S, that is, the graph over S in which there is an edge between two sensors $s_1, s_2 \in S$ if and only if the Euclidean distance between them, $d(s_1, s_2)$, is at most 1.

Assuming G is connected, our goal is to find a rooted spanning tree T of G. However, it is possible that some node u of the tree is faulty, in which case, in order not to lose the data at the children of u, we employ a data mule that visits each of u's children and collects the data from them. The mule's location is fixed when the gathering tree is determined; it is at one of the nodes of the tree. Then, if some node u is faulty, the mule must leave its base, travel to each of u's children and return to its base, where the mule can move freely in the plane. Thus, we would like to find a rooted spanning tree T of G and to determine the mule's location, such that the expected length of the mule's tour is minimum. In other words, our goal is to find a rooted tree T and a node v of T, such that the sum of $TSP^v(u)$, over all internal nodes u of T, is minimum, where $TSP^v(u)$ is the shortest tour beginning and ending at v and visiting each of the children of u.

This problem was introduced by Crowcroft et al. [2], who only studied its one-dimensional version. Subsequently, Yedidsion et al. [5] considered the two-dimensional version of the problem and presented an $O(n^2 \log n)$-time constant-factor approximation algorithm for it. That is, their algorithm finds a rooted tree and places the mule at one of its nodes, so that the sum of tours corresponding to their tree is bounded by a constant times the sum corresponding to the optimal solution. In this paper, we present an alternative, more efficient, constant-factor approximation algorithm for the (two-dimensional version of the) problem. The running time of our algorithm is $O(n \log n)$.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.

In general, some research has been done on various problems related to sensor networks that are augmented with mobile elements, see, e.g., [2–4]. See [5] for more details on related work.

2 Tree Construction

In this section we describe how to construct a spanning tree of the UDG induced by the set of sensors S. The constructed tree will have some desirable properties, mentioned below. In the subsequent section we will choose one of the nodes to serve as the tree's root (and as the mule's base); this will determine the direction of the edges of the tree.

We begin by laying a regular grid of edge length $\frac{1}{\sqrt{2}}$ over the input scene. (Notice that any two nodes within the same cell are at distance at most 1 from each other and are therefore connected by an edge in the underlying UDG.) Next, in each non-empty grid cell, we pick an arbitrary node to be the cell's *central node* (CN) and connect all other nodes in the cell to the CN. At this point the set of central nodes is a dominating set (DS) of UDG.

We now add some edges to connect between adjacent stars. More precisely, for any pair of stars, if the distance between them is at most one, i.e., if there exist a node u in one and v in the other such that $d(u, v) \leq 1$, then add an edge between the closest such pair of nodes. We do so by running the following algorithm.

```
1. For each cell $X$:
     Construct the Voronoi diagram of the nodes in $X$ and preprocess it for point location
       queries.
2. For each cell $X$:
     For each of the 24 cells $Y$ surrounding $X$ (i.e., for each cell in the first or second circle
       around $X$):
       a. For each node $u \in Y$:
            Find the node $v$ of $X$ which is closest to $u$.
       b. Let $(u, v)$ be the closest pair that was found.
       c. If $d(u, v) \leq 1$
            $E \leftarrow E \cup \{(u, v)\}$
3. Eliminate the cycles from the current graph by running DFS.
```

Let T be the tree that was obtained and notice that the set of its internal nodes is a connected dominating set (CDS) of UDG. We call an internal node of T a *backbone* node, and denote the set of internal nodes of T by BBN_T. Thus BBN_T is a CDS of UDG. Actually, BBN_T is an area constrained CDS (ACCDS) of UDG, where a CDS is an ACCDS if the number of nodes of the set in any disk of constant area A is $O(A)$. This is because the number of backbone nodes in any cell is bounded by 25 (the cell's CN plus at most 24 backbone nodes that are created by the code fragment above).

Finally, it is easy to see that the total time required to construct T is $O(n \log n)$. The total time spent on building the Voronoi diagrams and their corresponding search structures is $O(n \log n)$, and, for each node we perform a constant number of point location queries, so the total time spent on querying the diagrams is $O(n \log n)$.

3 Fixing the root and placing the mule

In [5], it was shown that the optimal solution with the additional constraint that the mule must be placed at the root is a 2-approximation of the unconstrained optimal solution. We therefore restrict our attention to the constrained version and present a constant-factor approximation for it, which in turn is a constant-factor approximation for the unconstrained version.

We focus on the more interesting and difficult case, where the area of the union of the unit disks around the sensors is greater than some constant. When this area is small, the number of backbone nodes (which is a linear function of the area) is small and the problem becomes much easier.

3.1 Placing the mule

Let T be the tree that was constructed in the previous section, and let $BBN_T = \{u_1, \ldots, u_m\}$ be the set of its internal nodes (i.e., backbone nodes). In this section, we describe how to determine the node of T in which the mule will be placed (and that will serve as the root of T). For a node u of T, let $w(u) = \sum_{i=1}^{m} d(u, u_i)$, that is, $w(u)$ is the sum of distances from u to the internal nodes of T. We would like to place the mule at the node of T for which this value is minimum, however we cannot afford to compute all these values. Instead, we choose a node v' whose value $w(v')$ is a good approximation of the minimum value.

We use the data structure of Bose et al. [1], which is built over a set of points in the plane (given some $\varepsilon > 0$), and which supports sum-of-distances queries. More precisely, we construct the data structure over the set BBN_T, in $O(\frac{1}{\varepsilon^2} n \log n)$ time and $O(\frac{n}{\varepsilon^2})$ space, and perform n queries in it, one per each node in T, in total $O(\frac{1}{\varepsilon^2} n \log n)$ time. The answer to a query with node u is a value $w_\varepsilon(u)$, such that $(1 - \varepsilon\sqrt{2})w(u) \le w_\varepsilon(u) \le (1 + \varepsilon\sqrt{2})w(u)$, and let v' be the node whose returned value (i.e., $w_\varepsilon(v')$) is minimum.

Denote by OPT_T the sum of tours corresponding to the optimal location in tree T, and denote by OPT_{T^*} the sum of tours corresponding to the optimal location in the (unknown) optimal tree T^*. We prove below that the sum of tours corresponding to v' is bounded by some constant times OPT_{T^*}.

▶ **Theorem 3.1.** *Choosing v' as the location for the mule yields a constant-factor approximation of OPT_{T^*}, for sufficiently large m. Moreover, the total running time of our algorithm is $O(n \log n)$.*

Proof. The proof is based on the following three claims, which correspond to Lemmas 3.2–3.5 below.

1. There exists a constant c' such that $OPT_T \le c' \cdot OPT_{T^*}$.
2. Placing the mule at the node v such that $w(v) = min\{w(u) : u \in BBN_T\}$ yields a 2-approximation of OPT_T, for sufficiently large m.
3. The sum of tours corresponding to v' is a 4-approximation of the sum of tours corresponding to v, for sufficiently large m.

From these claims it follows that placing the mule at v' yields a $c = 8c'$-approximation of OPT_{T^*}, for sufficiently large m. As for the running time, the tree T is constructed in $O(n \log n)$ time, after which the node v' is found in $O(n \log n)$ time. ◀

We now prove the 3 lemmas mentioned in the proof of Theorem 3.1.

▶ **Lemma 3.2.** *There exists a constant c' such that $OPT_T \le c' \cdot OPT_{T^*}$.*

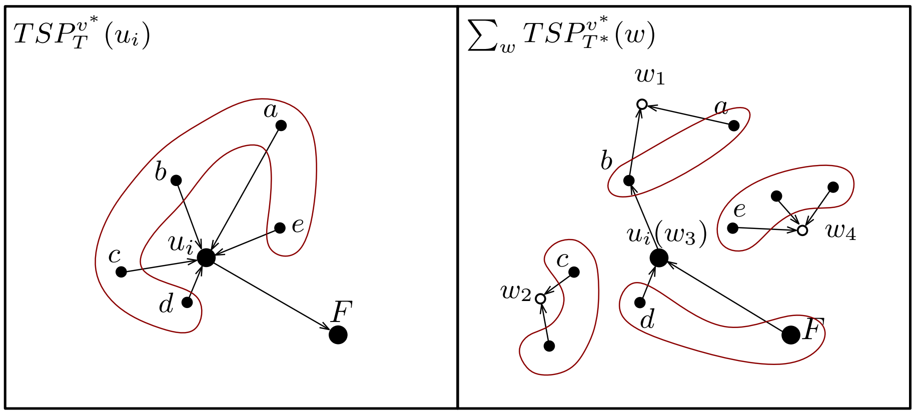

Figure 1 Proof of Lemma 3.2. Left: $C_T(u_i) = \{a, b, c, d, e\}$. Right: The nodes that T's mule will visit if u_i is faulty.

Proof. Let v^* be the optimal location in T^*. We will show that the sum of tours corresponding to v^* in T is already bounded by some constant c' times the sum of tours corresponding to v^* in T^*. But the former sum is at least OPT_T and the latter sum is equal to OPT_{T^*}, so we may conclude that $OPT_T \leq c' \cdot OPT_{T^*}$.

Let $C_T(u_i)$ be the set of u_i's children in T. Instead of visiting the nodes in $C_T(u_i)$ if u_i is faulty, T's mule will do the following. For each node w, which in T^* is a parent of a node in $C_T(u_i)$, T's mule will visit the nodes in $C_{T^*}(w)$. In other words, we replace $TSP_T^{v^*}(u_i)$ by $\sum_w TSP_{T^*}^{v^*}(w)$, where the sum is over all nodes w which in T^* have a child in $C_T(u_i)$; see Figure 1. Clearly, by doing so, T's mule will still visit the nodes in $C_T(u_i)$, but it may also visit other nodes, and in either way, the total distance traveled by T's mule can only increase.

Observe, however, that each of the nodes w in the latter sum is at distance at most 2 from u_i. So, since the internal nodes of T constitute an ACCDS, the number of internal nodes of T that lie within distance at most 2 from w is bounded by some constant c', and therefore, the number of times that T's mule will need to visit the nodes in $C_{T^*}(w)$ is bounded by some constant c'. We conclude that the total distance traveled by T's mule (after replacing the terms $TSP_T^{v^*}(u_i)$ by the corresponding sums) is bounded by $c' \sum_{w \in BBN_{T^*}} TSP_{T^*}^{v^*}(w) = c' \cdot OPT_{T^*}$, which implies that $\sum_{i=1}^{m} TSP_T^{v^*}(u_i) \leq c' \cdot OPT_{T^*}$. ◀

From now on, we are dealing only with the tree T, so we write TSP^v instead of TSP_T^v and BBN instead of BBN_T. Our proof of the next two lemmas relies on the following observation, which follows immediately from the way T was constructed.

▶ **Observation 3.3.** *Let v_1, v_2 be two nodes and let u be a backbone node. Then, there exists at most one node that is a child of u, when the mule is at v_1, but is not a child of u, when the mule is at v_2.*

▶ **Lemma 3.4.** *Placing the mule at the node v such that $w(v) = min\{w(u) : u \in BBN\}$ is a 2-approximation of OPT_T, for sufficiently large m.*

Proof. Let v^* be the node in which the mule is placed in the optimal solution for T, i.e., $\sum_{i=1}^{m} TSP^{v^*}(u_i) = OPT_T$. We first show that $\sum_{i=1}^{m} TSP^{v}(u_i) \leq \sum_{i=1}^{m}(TSP^{v^*}(u_i) + 6)$. Denote by s_i and t_i the first and last nodes that are visited in the tour taken by the mule based in v^* when u_i fails, and denote by $\pi(s_i, \ldots, t_i)$ the length of the portion of this tour beginning at s_i and ending at t_i. Then, $TSP^{v^*}(u_i) = d(v^*, s_i) + \pi(s_i, \ldots, t_i) + d(t_i, v^*)$, and, by Observation 3.3, $TSP^v(u_i) \le d(v, u_i) + d(u_i, s_i) + \pi(s_i, \ldots, t_i) + d(t_i, u_i) + d(u_i, v) + 2$, where 2 is an upper bound on the total length of the back-and-forth trips from u_i to visit the at most one child of v that is not also v^*'s child. So,

$$\begin{aligned}\sum_{i=1}^{m} TSP^v(u_i) &\le \sum_{i=1}^{m}(d(v,u_i) + d(u_i,s_i) + \pi(s_i,\ldots,t_i) + d(t_i,u_i) + d(u_i,v) + 2)\\ &\le^{(i)} \sum_{i=1}^{m}(d(v^*,u_i) + d(u_i,s_i) + \pi(s_i,\ldots,t_i) + d(t_i,u_i) + d(u_i,v^*) + 2)\\ &\le^{(ii)} \sum_{i=1}^{m}(d(v^*,s_i) + 1 + d(u_i,s_i) + \pi(s_i,\ldots,t_i) + d(t_i,u_i) + 1 + d(t_i,v^*) + 2)\\ &\le \sum_{i=1}^{m}(TSP^{v^*}(u_i) + 6)\ ,\end{aligned}$$

where inequality (i) is true since $w(v) \le w(v^*)$ and inequality (ii) is true since $d(v^*, u_i) \le d(v^*, s_i) + d(s_i, u_i) \le d(v^*, s_i) + 1$, and similarly, $d(u_i, v^*) \le d(u_i, t_i) + d(t_i, v^*) \le 1 + d(t_i, v^*)$.

Now, since BBN is an ACCDS, if we assume that m is sufficiently large, then the average tour length from v^* (i.e., $(\sum_{i=1}^{m} TSP^{v^*}(u_i))/m$) is greater than 6, and therefore $\sum_{i=1}^{m}(TSP^{v^*}(u_i) + 6) \le 2\sum_{i=1}^{m} TSP^{v^*}(u_i)$. Thus, for sufficiently large m, we get $\sum_{i=1}^{m} TSP^v(u_i) \le 2\sum_{i=1}^{m} TSP^{v^*}(u_i) = 2OPT_T$. ◀

▶ **Lemma 3.5.** $\sum_{i=1}^{m} TSP^{v'}(u_i) \le 4\sum_{i=1}^{m} TSP^v(u_i)$*, for sufficiently large* m*.*

Proof. Since $w_\varepsilon(v') \le w_\varepsilon(v)$,

$$(1-\varepsilon\sqrt{2})w(v') \le w_\varepsilon(v') \le w_\varepsilon(v) \le (1+\varepsilon\sqrt{2})w(v)\ ,$$

or

$$\sum_{i=1}^{m} d(v', u_i) \le \frac{1+\varepsilon\sqrt{2}}{1-\varepsilon\sqrt{2}} \sum_{i=1}^{m} d(v, u_i)\ .$$

Now, as in the proof of Lemma 3.4, we write $TSP^v(u_i) = d(v, s_i) + \pi(s_i, \ldots, t_i) + d(t_i, v)$, where s_i and t_i are the first and last nodes visited by the mule based at v when u_i fails, and $\pi(s_i, \ldots, t_i)$ is the portion of $TSP^v(u_i)$ beginning at s_i and ending at t_i. Then,

$$\sum_{i=1}^{m} d(v', s_i) \le \sum_{i=1}^{m}(d(v', u_i) + d(u_i, s_i)) \le \frac{1+\varepsilon\sqrt{2}}{1-\varepsilon\sqrt{2}} \sum_{i=1}^{m}(d(v, u_i) + d(u_i, s_i))\ ,$$

and, similarly,

$$\sum_{i=1}^{m} d(v', t_i) \le \frac{1+\varepsilon\sqrt{2}}{1-\varepsilon\sqrt{2}} \sum_{i=1}^{m}(d(v, u_i) + d(u_i, t_i))\ .$$

Again, as in the proof of Lemma 3.4,

$$\sum_{i=1}^{m} TSP^{v'}(u_i) \le \sum_{i=1}^{m}(d(v', s_i) + \pi(s_i, \ldots, t_i) + d(t_i, v') + 2)\ ,$$

so

$$\begin{aligned}\sum_{i=1}^{m} TSP^{v'}(u_i) &\le \frac{1+\varepsilon\sqrt{2}}{1-\varepsilon\sqrt{2}} \sum_{i=1}^{m}(d(v,u_i) + d(u_i,s_i) + \pi(s_i,\ldots,t_i) + d(v,u_i) + d(u_i,t_i) + 2)\\ &\le \frac{1+\varepsilon\sqrt{2}}{1-\varepsilon\sqrt{2}} \sum_{i=1}^{m}(TSP^v(u_i) + 6)\ .\end{aligned}$$

Now, since BBN is an ACCDS, if we assume that m is sufficiently large, then the average tour length from v (i.e., $(\sum_{i=1}^{m} TSP^{v}(u_i))/m$) is greater than 6, and therefore $\sum_{i=1}^{m}(TSP^{v}(u_i)+6) \leq 2\sum_{i=1}^{m} TSP^{v}(u_i)$. Thus, for sufficiently large m, we get $\sum_{i=1}^{m} TSP^{v'}(u_i) \leq 4\sum_{i=1}^{m} TSP^{v}(u_i)$, by choosing $\varepsilon < 1/(3\sqrt{2})$. ◀

Acknowledgements

The author wishes to thank Matya Katz for his help during all stages of this work.

References

1 Prosenjit Bose, Anil Maheshwari, and Pat Morin. Fast approximations for sums of distances, clustering and the fermat-weber problem. *Comput. Geom.*, 24(3):135–146, 2003.

2 Jon Crowcroft, Liron Levin, and Michael Segal. Using data mules for sensor network data recovery. *Ad Hoc Networks*, 40:26–36, 2016.

3 M. Di Francesco, S. K. Das, and A. Giuseppe. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 8.1:7–38, 2011.

4 O. Tedas, V. Isler, J. h. Lim, and A. Terzis. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16.1:22, 2009.

5 Harel Yedidsion, Aritra Banik, Paz Carmi, Matthew J. Katz, and Michael Segal. Efficient data retrieval in faulty sensor networks using a mobile mule. In *15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2017, Paris, France, May 15-19, 2017*, pages 1–8, 2017.