The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

# Data Gathering Tours for Mobile Robots

Deepak Bhadauria and Volkan Isler

***Abstract*— We study a path planning problem which arises when multiple robots are used to gather data from stationary devices with wireless communication capabilities. Each device has a given communication range, and stores a fixed amount of data. The objective of the robots is to gather the data from these devices and to upload it to a base-station/gateway. We introduce a new optimization problem called the Data Gathering Problem (DGP). In DGP, the objective is to compute a tour for each robot in such a way that minimizes the time to collect data from all devices. In order to download the data from a device, a robot must visit a point within the communication range of the device. Then, it spends a fixed amount of time to download the data. Thus, the time to complete a tour depends on not only the travel time but also the time to download the data, and the number of devices visited along the tour.**

**First, we study a special case of DGP where the robots' motion is restricted to a curve which contains the base station at one end. Next, we study the 2D version. We show that two existing algorithms for variants of the Traveling Salesperson Problem can be combined and adapted to obtain a constant factor approximation to DGP. Afterwards, we present an improvement for sparse deployments. We also present simulations which shed light on the utility of data gathering using mobile robots.**

## I. Introduction

There are many applications Wireless Sensor Network (WSN) applications in which a large area must be covered with a sparse deployment. For example, in habitat monitoring, soil humidity and temperature at areas visited by a particular species is collected. In order to use the current WSN technology in such applications, a dense network must be deployed. This is because WSN nodes typically have short communication ranges, and many nodes must act as relays so that the data can be gathered at a base station.

In our recent work, we presented an alternative to static relay-nodes for such applications: using robots as data mules to collect the data from sensors [8]. This approach has a number of advantages over deploying a dense static network. First of all, since relay nodes are no longer needed, operational costs are minimized. Second, the lifetime of the network is maximized because the robots can get close to the sensor nodes to download the data. In addition to reducing the energy consumption during transmission (less power is needed), proximity also reduces the data loss rate, which results in smaller number of transmissions per byte. In [8], we experimentally demonstrated the utility of a robotic mule system for gathering data.

The authors with the University of Minnesota. Emails: {bhadau,isler}@cs.umn.edu

This work was supported by the grants NSF CCF-0907658 and NSF IIS-0917676.

In the present work, we address the problem of planning the routes of robotic mules: given locations of $n$ sensors, compute the routes of $\kappa$ robots so that the time to download the data from all sensors is minimized. Throughout the paper, we will refer to this problem as the *Data Gathering Problem ($\kappa$-DGP)*. Note that in DGP, the cost incurred by a robot depends on not only the robot's travel time but also the time to download data from a sensor, and the number of sensors assigned to the robot.

The well-known Traveling Salesperson Problem (TSP) asks for the shortest path for a salesperson to visit $n$ cities [1]. There is a variant of TSP, called $k$-TSP, where $k$ travelers visit $n$ cities, and the objective is to minimize the length of the maximum tour [4]. Even though DGP resembles TSP, a closer look reveals important differences as shown in Figure 1.

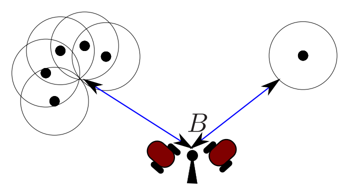


Fig. 1. Two robots charged with collecting data from the sensors and relaying them to the base station $B$. The filled circles correspond to sensor locations. The circle around a sensor illustrates its communication range. The figure shows optimum TSP tours for the two robots which minimize the maximum travel distkance by any robot. This solution is not appropriate for data gathering because the robot assigned to the left group would spend a lot of time downloading the data from the sensors.

As another example, consider a special case where all sensors and the base station are on a line. Imagine that all sensors are to the right of the base station. In TSP, there is no utility in employing more than one robot for this case: the robot that will visit the furthest sensor can visit all other sensors on the way. However, when the download time is incorporated, the utility of employing multiple robots becomes evident.

Another aspect where DGP differs from TSP is due to the presence of a non-zero communication range. As shown in Figure 1, the robot does not need to visit the precise location of a sensor. Instead, it needs to visit a point in its acceptable communication range[1] to download the data. There is a variant of TSP, called TSP with Neighborhoods (TSPN) which captures this aspect of DGP. In the geometric version of TSPN, we are given $n$ disks. The objective is to

[1]This is an application dependent parameter that depends on the characteristics of the signal, environment and acceptable signal quality and energy consumption levels.

978-1-4244-3804-4/09/$25.00 ©2009 IEEE

compute the shortest tour that visits at least one point in each disk. Even though efficient algorithms for TSPN exist [2], [6], there are no algorithms for $k$-TSPN where the objective is to compute TSPN tours for $k$ robots.

*Our results and organization of the paper:* In Section III, we formalize the Data Gathering Problem. In this paper, we make three main contributions. First, in Section IV, we present an optimal algorithm for the 1D version where the robots are restricted to move along a curve which contains a base station at the starting point. For the 2D case (Section IV), we show that the $k$-TSP algorithm presented in [4] can be combined with the TSPN algorithm of [2], and the resulting algorithm can be modified to obtain a constant-factor approximation algorithm for DGP in 2D (Section V-B). Next, we focus on sparse deployments where the utility of using robots is significant, and present an improved algorithm for this case in Section V-C. In Section VI, we present simulations which provide further insights on the use of robots for gathering data. In the next section, we start with a brief overview of the related work.

## II. Related Work

The Traveling Salesperson Problem (TSP) is a fundamental, widely studied optimization problem [1]. The work presented in this paper is based on algorithms for two variants of TSP. The TSP with Neighborhoods [2], and $k$-TSP [4]. An overview of these two algorithms is presented in Section V-A.

Exploiting mobility in collecting sensor data has received some recent attention. For example, in [7], Shah et al presented an architecture that uses mobile entities in the environment for data delivery. However, in most of the related literature, mobility is treated as an uncontrolled process. A recent review on the state of the art in exploiting sink mobility can be found in [5].

Yuan et al. formulate the problem of collecting sensor data using a single robot as a TSPN instance [10]. They do not address the time spent in downloading data. As shown in Figure 1, ignoring transmission time can worsen the performance of the system drastically. Tirta et al. presented algorithms to schedule visits of a mobile agent to collect data from cluster heads [9]. The authors present heuristics which focus on data latency and data aggregation rate of clusters.

In [3], Dunbabin et al. present an underwater data muling system. In the underwater scenario, sensors and underwater vehicles communicate through optical communication, which requires a close proximity as well as good view-angle to start the communication. Moreover, since GPS localization is not available under the water, the vehicle has to navigate under high localization uncertainty. This makes designing global routing algorithms challenging. The authors propose a solution where the vehicle performs a spiral movement to find the sensors. This strategy is not efficient for our scenario, in which the sensor locations are known and the robot can localize itself.

## III. Problem definition

Suppose we are given the locations of $n$ identical sensors. There is a base station $B$, and $\kappa$ robots which can navigate in the environment are charged with downloading the data from each sensor and uploading it to the base station. We define *coverage of a sensor* as receiving all data from that sensor and transmitting it to the base station. We make assumptions as given below.

The sensors are identical. We assume that each sensor can sense data and transmit it up to a distance $T_r$ units with uniform data rate. Therefore, the time required to download data from every sensor, $T_d$, is identical.

The data is downloaded by $\kappa$ identical robots which have wireless communication capabilities and can travel at a constant speed of $\nu$. In this work, we do not consider higher order constraints such as acceleration. The robots have infinite buffer capacities (since they can carry large storage devices). Similarly, we do not consider energy limitations for robots. We assume that the robots can localize themselves and navigate in the environment.

Even though there may be obstacles in the environment, we assume that the communication disks are obstacle-free. That is, no obstacle intersects the communication disk of any sensor.

## IV. The 1-D Data Gathering Problem.

In this section, we study the 1-D version of the data gathering problem where the robots are restricted to move along a curve $X$. This case has practical applications in scenarios where robots move along a rail-line, or there is a single path they can move along in a rough terrain.

We assume that base station is at an end point of the curve $X$; i.e. at $x = 0$ where $x$ is the parameterization of the underlying curve.

For each sensor $s$, compute the intersection the communication disk (centered at $s$ with radius is $T_r$) with the curve $X$. Suppose all intersections are on one side of the base-station.

Let $x_s$ be the first point of this intersection along $X$. We will choose $x_s$ as the *download location* of a sensor $s$. Since all intersections are assumed to be on one side of the base station, any robot which will download data from $s$ can do so from location $x_s$ without incurring additional travel costs. Hereafter, we represent each sensor with its download location. For this version of the problem where $x_s > 0$ for all $s$, we will present an optimal algorithm for gathering data with $\kappa$ robots.

Consider a solution to the 1D data gathering problem. Let $U = \{u_1, u_2, \dots, u_k\}$ and $V = \{v_1, v_2, \dots, v_l\}$ be the sets of sensors assigned to robots $u$ and $v$ (we overload the notation and use $u_i$ to refer both a sensor and its download location) ordered and labeled such that $u_1 \le u_2 \le \dots \le u_k$ and $v_1 \le v_2 \le \dots \le v_l$. While ordering download locations we break ties arbitrarily. Without loss of generality, assume that $u_1 \le v_1$. We say that $u$ and $v$ are *non-overlapping* if $u_k \le v1$.

The following lemma sheds light onto the structure of optimal data collection.

*Lemma 4.1:* There exists an optimal solution to cover $n$ sensors with $\kappa$ robots in which every pair of robots is non-overlapping.

*Proof:* Among all optimal solutions, consider the optimal solution $S$ with the largest number of non-overlapping pairs of robots. We claim that no pair of robots in $S$ is overlapping.

Suppose, to the contrary, that there is a pair of robots $u$ & $v$ in $S$ which are overlapping. Let $U = \{u_1, u_2, \ldots, u_k\}$ and $V = \{v_1, v_2, \ldots, v_l\}$ be the sets of sensors assigned to $u$ and $v$.

The sensors in $U$ and $V$ can overlap in two primary ways shown in Figure 2.

Case (a): When there is a partial overlap. Let the coverage time taken by $u$ and $v$ be $T(u)$ and $T(v)$.

$$T(u) = \frac{u_k}{\nu} + kT_d \quad \text{and} \quad T(v) = \frac{v_l}{\nu} + lT_d$$

We reassign sensor $u_k$ to robot $v$ and sensor $v_1$ to robot $u$. New sets of sensors $U' = U - \{u_k\} + \{v_1\}$ and $V' = V - \{v_1\} + \{u_k\}$ are assigned to $u$ and $v$. Let $u'_k$ be the sensor in $U'$ which is farthest from base station. New coverage times are:

$$T'(u) = \frac{u'_k}{\nu} + kT_d \quad \text{and} \quad T'(v) = \frac{v_l}{\nu} + lT_d$$

Since $u'_k \leq u_k$, we have $T'(u) \leq T(u)$. Also $T'(v) = T(v)$. The reassignment operation does not increase the coverage cost of any of the two robots. Continue reassigning in the similar way until $u'_k \leq v_1$.

Case (b): When there is a complete overlap. In this case reassign sensor $u_1$ to robot $v$ and sensor $v_1$ to robot $u$. This does not increase the cost of coverage of by any of the robot $u$ and $v$. Now the case is similar to case (a) but with robots swapped. This case can be dealt in a similar way as (a).

Let $S'$ the solution obtained by reassignment of $S$. $S'$ is at most as costly as $S$ and has one more pair of non-overlapping robots. This contradicts the maximality of $S$. Hence, $S$ has no overlapping pair of robots. ∎

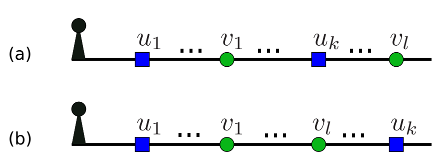


Fig. 2. Data collection on the line: (a) partial overlap (b) complete overlap.

Lemma 4.1 sheds light onto the structure of sensor assignments in an optimal solution. We now present a dynamic programming algorithm to exploit this structure and to gather the data from $n$ sensors using $\kappa$ robots in an optimal fashion.

Let us order and label sensors from $s_1$ to $s_n$ with increasing distance of their download locations from base station (again, $s_i$ refers to both the $i^{th}$ sensor and its download location). Let $cost(k, l)$ be the cost to cover sensors $s_k$ to $s_l$, $k \leq l$ by a robot. Therefore $cost(k, l) = \frac{s_l}{v} + (l - k + 1)T_d$. We create an $n \times \kappa$ table $T$. Each entry $T[i, j]$ represents the optimal cost to cover $s_1$ to $s_i$ sensors by $j$ robots. The table is computed using the following recurrence equation:

$$T[i,j] = \begin{cases} \frac{s_i}{v} + iT_d & \text{if } j = 1 \\ \min_{k \in \{1, j-1\}} (T[k, j-1] + cost(k, i)) & \text{otherwise} \end{cases} \tag{1}$$

*Lemma 4.2:* $T[i, j]$ will give us the optimal coverage time for the first $i$ sensors using $j$ mules.

*Proof:* We prove the lemma by induction on $j$, the number of robots. *Basis:* For $j = 1$ the minimum time to cover $s_1$ to $s_i$, $T[i, 1] = \frac{s_i}{v} + iT_d$ for all $i \in \{1, n\}$. This is minimum because to cover $i$ sensors any robot has to travel up to download location of farthest sensor and download data from all the sensors. *Induction hypothesis:* Let $T[l, j-1]$ be minimum time required to cover first $l$ sensors using $j - 1$ robots $\forall l \in \{1, n\}$. Now for $j$ robots $T[i, j] = \min(T[l, j-1] + cost(l+1, i))$ where $l \in \{1, i-1\}$. Since $T[l, j-1]$ is minimum coverage time for $l$ sensors with $j - 1$ mules and the value of $T[i, j]$ is set to minimum of all $i - 1$ possible values, $T[i, j]$ is minimum coverage time of $i$ sensors with $j$ mules. ∎

Thus, the entry $T[n, \kappa]$ gives us the desired solution. The running time of the algorithm can be easily seen to be $O(n^2\kappa)$. The main result of this section is summarized by the following theorem.

*Theorem 4.3:* There exists an optimal polynomial time algorithm to solve the 1-D version of the data gathering problem when all the sensors are on one side of the base station.

## V. K-ROBOT COVERAGE IN 2D

We start this section by reviewing algorithms for two relevant TSP variants: In Section V-A, we present an overview of a constant factor approximation algorithm by Dumitrescu and Mitchell [2] for TSP with Neighborhoods where the neighborhoods are uniform disks, as well as a constant factor k-TSP algorithm. In Section V-B, we show how these two algorithms can be modified and combined to obtain a constant factor algorithm for the data-gathering problem. Finally, in Section V-C, we present an algorithm which gives improved results when the sensors are "sparse"[2]

### A. TSPN tour algorithm and TSP splitting algorithm

In [2], Dumitrescu and Mitchell present a constant-factor approximation algorithm for TSPN with uniform disk neighborhoods. The approximation ratio of the algorithm which we refer to as *TSPN_TOUR* is 11.15.

TSPN_TOUR first finds out a maximal independent set (MIS) of non-intersecting disks. Then it creates a TSP tour $T_I$ which visits the center of each disk in MIS. A TSPN tour is formed from the TSP tour as follows: the TSPN tour starts from the intersection of an arbitrary disk in MIS and $T_I$. It then follows $T_I$ in clockwise direction. If the boundary of a MIS disk $D$ is encountered, $D$ is traversed clockwise along the boundary until the next intersection of $T_I$ with $D$. This

[2]We will quantify the notion of sparsity in Section V-C as well.

continues until the tour returns to starting point. Now the $T_I$ is traversed in similar fashion but in counter clockwise direction until start point is encountered. Fig 3 illustrates a TSPN tour obtained by the algorithm TSPN_TOUR.

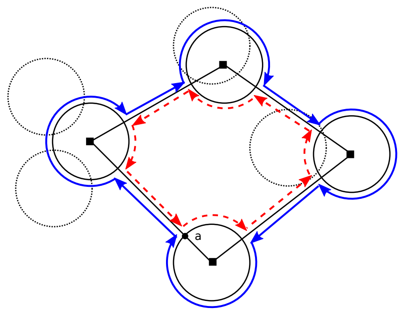


Fig. 3. A TSPN tour constructed by algorithm TSPN_TOUR. The disks in MIS are drawn with heavy solid lines. The tour contains two subtours. After visiting a circle, one subtour traverses its boundary clockwise until the next intersection of the disk with the tour. On the way back, the other tour traverses the disk counter-clockwise.

Frederickson et al [4] present an algorithm, *k-SPLITOUR*, to split a TSP tour into $k$ subtours. In this algorithm one travels along the TSP tour and the tour is split each time when the cost reaches a threshold which is decided by the average cost of a $k$-subtour. This algorithm gives an approximation bound of $\frac{5}{2} - \frac{1}{k}$ for $k$-TSP.

In the next section, we show how these two algorithms can be combined to solve the data gathering problem in 2D.

### B. Algorithm for DGP in 2D

The main idea of the algorithm is to create a TSPN tour of sensors and divide that tour into $\kappa$ subtours such that each subtour is of almost similar cost. The main steps of the algorithm are as follows:

1) For $n$ sensors and base station $b$ find a TSPN tour $R = (b, s_1, s_2, ..., s_n, b)$ with $cost(R) = \tau_1$, where $s_i$ is a sensor node and $\tau_1$ is the coverage time of $n$ sensors by one robot.
2) For each $j, 1 \leq j < \kappa$ find the last sensor $s_{n_j}$, $n_j \in \{1, n\}$ such that the cost (time to travel plus download data) of path from $b$ to $sn_j$ along $R$ is not greater than $(j/k)(\tau_1 - 2c_{max}) + c_{max}$. Here $c_{max}$ is the time taken for a robot to travel from base station to the download location farthest from base station.
3) Let $s_i^j$ represents the $i^{th}$ sensor in $j^{th}$ subtour. Obtain the $\kappa$ subtours by forming $j^{th}$ subtour as $R_j = (b, s_1^j, s_2^j, ..., s_{n_j}^j, b)$ for all $j \in \{1, \kappa\}$. Note that $s_{n_{j-1}}^{j-1} = s_1^j$ for $j \in \{1, \kappa - 1\}$ and $s_{n_\kappa}^\kappa = s_n$.

We now show that by combining the two algorithms for TSPN and k-TSP, one obtains a constant factor algorithm for the data gathering problem.

*Theorem 5.1:* If $\tau_k$ is the cost of the largest subtour generated by algorithm and $\tau_k^*$ is cost of the largest subtour in the optimal solution for the data gathering problem, then

$$\tau_k/\tau_k^* \leq e + 2 - 1/k \tag{2}$$

where $e$ is the approximation ratio of the algorithm used to find TSPN tour at step 1 of DGP algorithm.

*Proof:* We sketch the main steps in the proof which parallels the proof of the performance of the *k-SPLITOUR* algorithm by [4]. The cost of any tour $R_i$, $1 \leq i \leq k$, does not exceed $(1/k)(\tau_1 - 2c_{max}) + 2c_{max}$.

Therefore

$$\tau_k = \max(cost(R_i)) \leq (1/k)(\tau_1 - 2c_{max}) + 2c_{max} \tag{3}$$

By triangle inequality $\tau_k^* \geq (1/k)\tau_1^*$ where $\tau_1^*$ is the optimal cost of coverage with 1 robot. This can be proven by contradiction: Let us assume that $\tau_k^* < (1/k)\tau_1^*$. We can combine the subtours in such a way that the last sensor of each subtour is connected to the first sensor of the next subtour. By the triangle inequality the new edge will be shorter than the sum of the edges deleted (edge from last sensor of the subtour to the base station and edge from the first sensor of the next subtour to the base station). But this means $\tau_1^*$ is not optimal which is a contradiction.

Let the cost of TSPN tour at step 1 of the algorithm with $T_d = 0$ (Cost of a "regular" TSPN tour) be $\tau$ and the cost of optimal TSPN tour with $T_d = 0$ be $\tau^*$. Depending on implementation a robot may download some amount of data while it is traveling inside the disk (i.e. overlap time of traveling and downloading) of a sensor. We have $\tau_1 \leq \tau + nT_d$. Also $\tau_1^* \geq \max(\tau^*, nT_d)$. So we get

$$\frac{\tau_1}{\tau_1^*} \leq \frac{\tau + nT_d}{\tau_1^*} \leq \frac{\tau}{\tau_1^*} + \frac{nT_d}{\tau_1^*} \leq e + 1 \tag{4}$$

where $e$ is the approximation ratio of the algorithm used to find TSPN tour. Combining the results from Equation 3, triangle inequality, Equation 4 and from the fact that $c_{max} \leq \frac{1}{2}\tau_k^*$ we get $\tau_k/\tau_k^* \leq e + 2 - 1/k$. ■

If TSPN_TOUR is used for finding TSPN tour for step 1 of the above algorithm we have to be careful in dividing the tour. This is because of the fact that the last sensor covered in a subtour may be the one whose disk is not in MIS. For such cases we need to know the exact point on the disk boundary where we have to make the split. Let $A$ be a disk in MIS and $B$ be another disk intersecting $A$. To cover sensor at $B$, a robot stops at the point where the line joining center of $A$ and the center of $B$ intersects with boundary of disk $A$ for downloading data from $B$. After downloading data from sensor at $B$, the mule continues its TSPN tour. Note that data from sensor at $A$ is downloaded from the point where tour first meets boundary of $A$. By fixing the download location of sensors whose disk intersect with $A$ it becomes easy to divide the tour. Fig 4 shows one such subtour division. Note that second subtour does not visit the base station when it starts the tour in anti-clockwise direction.

### C. Improvement for sparse sensor networks

In TSPN_TOUR algorithm each of the edges connecting the pairwise disjoint disks is traversed twice. This can be costly when the disks are far apart which is the case for sparse sensor networks. For this case, we present an improved algorithm which constructs the TSPN tour differently

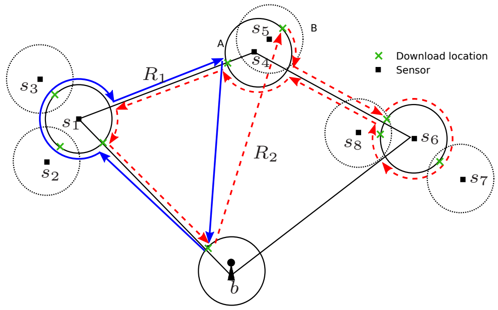


Fig. 4. Division of a TSPN tour into 2 subtours using DGP algorithm.

than TSPN_TOUR. Afterwards, we formalize the notion of sparsity and provide the condition on which it will be less costly than TSPN_TOUR. We refer to this method of construction as *ONE_WAY_TSPN_TOUR*.

We justify the utility of *ONE_WAY_TSPN_TOUR* in simulations where we show that it yields considerable cost reduction for sparse networks.

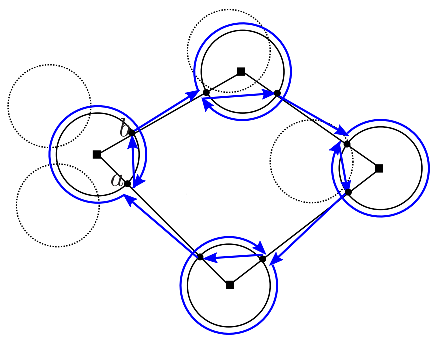


Fig. 5. ONE_WAY_TSPN_TOUR makes a complete tour after visiting each disk in the MIS. This approach improves the coverage time when there are sensors which are far apart.

In *ONE_WAY_TSPN_TOUR*, we start the TSPN tour similarly as in TSPN_TOUR, i.e., from the point of intersection of boundary of disk of an arbitrary vertex and $T_I$. Traverse along $T_I$ until the point of intersection, $a$, of the boundary of an MIS disk is encountered. Make a complete tour of the disk boundary until $a$ is encountered again. Now, from $a$ traverse directly to the next point of intersection, $b$, of $T_I$ and this disk (Figure 5). From $b$ continue along $T_I$ in similar fashion until the point from where we started is reached. Let the cost of $T_I$ in terms of distance be $C_I$ and the number of disks in MIS be $m$. We can compare TSPN_TOUR with ONE_WAY_TSPN_TOUR as follows. When compared with ONE_WAY_TSPN_TOUR, TSPN_TOUR covers an extra distance of $C_I - 2mT_r$, since it traverses the tour twice. On the other hand, ONE_WAY_TSPN_TOUR may cover an extra distance of $2mT_r$ (along the diameter of the disk to get back on the tour). Therefore, we can use ONE_WAY_TSPN_TOUR whenever $2mT_r \leq C_I - 2mT_r$. This gives the condition

$$T_r \leq \frac{C_I}{4m} \tag{5}$$

When this condition is satisfied we will prefer ONE_WAY_TSPN_TOUR over TSPN_TOUR because it gives at least $C_I - 4mT_r$ saving in distance to travel. We believe that this improvement will be significant in environmental monitoring applications where clusters of sensors are sparsely deployed over large areas. In figure 6, we present an instance of DGP where the proposed modification yields significant improvements.

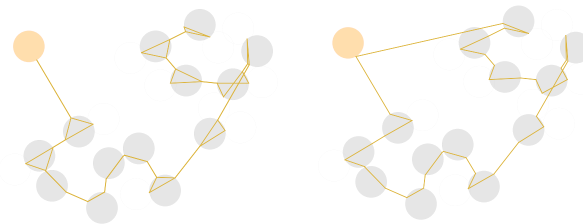

Fig. 6. Data gathering tours for a single robot based on **Left:** TSPN_TOUR (the total coverage time is 3829 units), **Right:** ONE_WAY_TSPN_TOUR (the total coverage time is 2612 units). In this instance, the proposed modification yields significant improvements. In both figures, gray disks correspond to the disks in MIS. The base station is the yellow disk on the top left.

## VI. Simulations and Further Insights

In this section, we further study DGP with simulations. In the first experiment, we investigate the utility of increasing the number of robots. In Figure 7, we plot the coverage time as a function of $\kappa$, the number of robots. In this experiment, we placed 100 sensors uniformly at random in a $600 \times 600$ environment. The communication radius $T_r$ was chosen to be 30. As the figure shows there is a steep decrease in cost as the number of robot increases. As the number of robots approach the number of independent disks the decrement in cost is lesser.

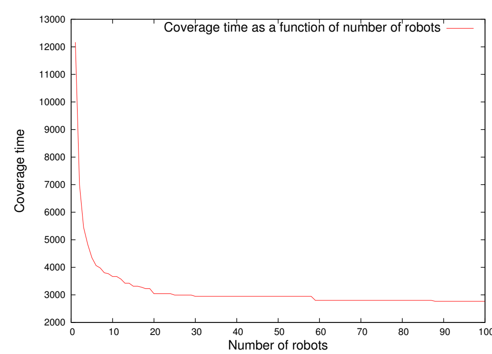


Fig. 7. The coverage time as a function of robots for 100 sensors deployed uniformly at random in a $600 \times 600$ area.

In TSPN-based partitioning, we compute a single TSPN tour which is then divided among the robots. When the sensors are uniformly deployed, a reasonable alternative is to divide the environment into $\kappa$ equal regions and to assign a robot to each region. After this assignment, the robots can compute TSPN tours for the sensors in their region. In the next experiment, we compare these two approaches.

Figure 8 shows the histogram of the ratio of the coverage time of area-based partition to the coverage time of the TSPN based partition. To obtain the histogram, we performed 100 trials in a $600 \times 600$ environment with uniformly placed sensors. The number of sensors in each trial was 100. In area based partitioning we divided the enivronment into six $300 \times 200$ regions and covered each region by a single robot. For uniform deployment, the performance of the two algorithms was comparable. On the average, TSPN based partition was only 1.03 times better. The highest ratio was 1.21 (i.e TSPN based partition was 21% better.) The tours for this instance are shown in Figure 9. The lowest ratio (where area based partitioning was better) was 0.87.

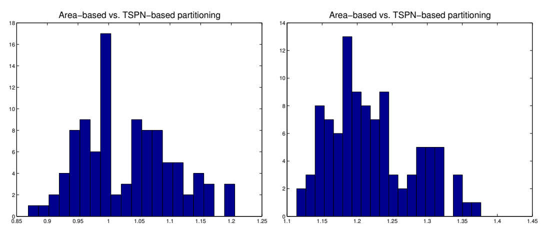


Fig. 8. The histograms shows the ratio of the coverage time of an area-based partitioning algorithm to the coverage time of the TSPN-based partitioning algorithm.**Left:** In this case the sensors were deployed uniformly at random, and the performance of the two algorithms was comparable. **Right:**In this case the sensors were deployed non-uniformly(See Figure 10) and the TSPN-based partitioning approach clearly outperformed the area-based approach.

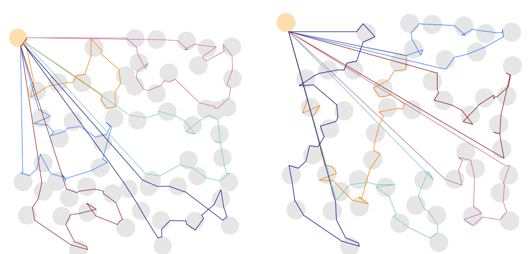

Fig. 9. **Left:** Data gathering using an area-based partitioning approach. **Right:** Data gathering using a TSPN-based partition. In this instance, TSPN-based partitioning was 21% better. Gray disks correspond to the disks in MIS for each partition. The base station is the yellow disk on the top left.

However, when the distribution of the sensors in not uniform, TSPN-based partition outperforms area-based partition. In the next experiment, we deployed 100 sensors in a $600 \times 600$ environment. The distribution of the sensors were denser in the upper right and lower left portions of the environment (Figure 10). The histogram of the ratio in 100 experiments clearly shows that TSPN-based partitioning outperforms area-based approach (Figure 8). On the average, TSPN based partition was 1.22 times better. The highest ratio was 1.37 (i.e TSPN based partition was 37% better.) The tours for this instance are shown in Figure 10. The lowest ratio was 1.11 (i.e. in all instance TSPN based partition outperformed area-based).

## VII. Concluding Remarks

In this paper, we introduced a new path planning problem, the Data Gathering Problem (DGP), which arises in

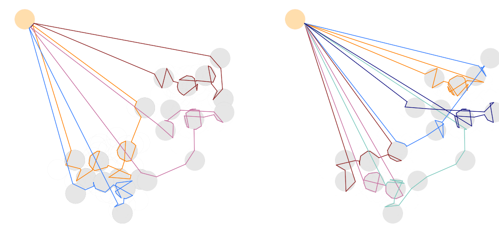

Fig. 10. **Left:** Data gathering using an area-based partitioning approach. **Right:** Data gathering using a TSPN-based partition. In this instance, TSPN-based partitioning was 37% better. In both figures, gray disks correspond to the disks in MIS. The base station is the yellow disk on the top left.

scenarios where robots act as *data mules* to download data from stationary wireless devices. We presented an optimal, polynomial-time algorithm for a special case where the robots are restricted to move along a curve which contains the base station at one end. For the 2D version, we showed that two algorithms developed for variants of the TSP problem can be combined and adapted to obtain a constant factor approximation algorithm for DGP in 2D. We also presented an improvement for sparse networks where robots spend significant time to travel between clusters that are far away. This is one of the scenarios where the utility of using robots for data collection is evident.

Our future work includes extending our work to heterogeneous devices with varying communication ranges and environments with obstacles. The algorithms presented in the paper can accommodate obstacles as long as they do not intersect with the communication disks. When this happens, a new algorithm for DGP in the presence of obstacles is needed.

## References


[1] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[2] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for tsp with neighborhoods in the plane. *J. Algorithms*, 48(1):135–159, 2003.

[3] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. *ICRA*, pages 2091–2098, 15-19, 2006.

[4] G. Frederickson, M. Hecht, and C. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[5] J. Ma, C. Chen, and J. Salomaa. mwsn for large scale mobile sensing. *J. Signal Process. Syst.*, 51(2):195–206, 2008.

[6] J. S. B. Mitchell. A ptas for tsp with neighborhoods among fat regions in the plane. In *SODA '07*, pages 11–18, 2007.

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

[8] O. Tekdas, J.H. Lim, A. Terzis, and V. Isler. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 2008. Accepted to the Special Issue on Wireless Communications in Networked Robotics.

[9] Y. Tirta, Zhiyuan Li, Yung-Hsiang Lu, and S. Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. *ICCCN*, pages 515–519, Oct 2004.

[10] B. Yuan, M. Orlowska, and S. Sadiq. On the optimal robot routing problem in wireless sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(9):1252–1261, 2007.