PATH PLANNING ALGORITHMS FOR ROBOTS IN A DATA MULING SYSTEM

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

DEEPAK BHADAURIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

IBRAHIM VOLKAN ISLER

December, 2010

Acknowledgements

There are many people that have earned my gratitude for their contribution to my time in graduate school. First and foremost, my deepest gratitude goes to my advisor Prof. Volkan Isler. I would like to thank him for giving me the opportunity to become a part of Robotic Sensor Networks (RSN) Lab. His advice has been extraordinary by all means, dedicating extensive amounts of time to help me in research, improving my technical writing and oral presentation skills and gearing me ready for challenges in life. His enthusisasm for research and knack for learning inspires me most, encourages me to have a learning mind at every stage of life.

The work in this thesis is a result of a very fruitful side-by-side collaboration with Onur Tekdas. Not only he deserves my gratitude for sharing with me his invaluable research experience, but significant credit due to his contribution to the work reported here. The results of this thesis are further benefited from discussions with current members of the RSN group especially Nikhil Karnad and Pratap Tokekar. I thanks all RSN members for their contribuitions.

My sincere gratitude is also due to my family and friends, for their continuous generosity in supporting me, for their precious time and sincerity to care about me, for their pleasant company and sharing of those exciting moments. Especially to my parents, our love is always the essential strength for us to face up everything coming to us and rejoice upon everything rejoicing us.

Abstract

We study two path planning problems that arise in data muling systems where robots are charged to collect data from wireless devices dispersed across a large environment. In such applications, deploying a network of stationary wireless sensors may be infeasible when many relay nodes are deployed to ensure connectivity. Instead, a few robots can be used as data mules to collect data from these devices.

The first problem studied in this thesis is to find tours for multiple robots so as to collect data from all sensors in the least amount of time. We refer to this problem as the Data Gathering Problem (DGP). We assume that sensors have a uniform disk communication model. In this model, data can be downloaded from a sensor with fixed rate inside its communication disk. We present an optimal algorithm for the one dimensional version of DGP. For the two dimensional version we present a constant factor approximation algorithm. Afterwards, we present field experiments in which an autonomous robotic data mule collects data from sensor nodes deployed over a large area.

Next, we study data collection problem with a more realistic communication model for sensors. In experiments we found that the time taken to download data from a sensor s is a function of the locations of the robot and s: If the robot is a distance r_{in} away from s, it can download the sensor's data reliably in T_{in} units of time. If the distance is greater than r_{in} but less than r_{out} , robot can still download data but due to higher packet loss probability the average download time T_{out} is higher ($T_{out} > T_{in}$). We refer to this model as the Two-Ring communication model and the corresponding path planning problem as the Two-Ring Tour (TRT) problem. We present a constant factor approximation algorithm for the general case. The algorithm uses a polynomial time approximation scheme as a subroutine. Though the scheme has polynomial running time, its running time is impractically large. It is also very complex to implement. Therefore we study special cases of the TRT problem and present efficient algorithms for them.

For robotic data mules to be useful, the robots must be capable of operating in the field for extended periods of time. Therefore, in the last part of the thesis we initiate

the study of solar energy harvesting for robotic navigation. Our primary contribution is an experimental model of energy consumption and harvesting as a function of environmental parameters. We demonstrate the utility of this model in a simple navigation task.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A wireless sensor network (WSN) comprises of a network of sensing devices which have the capability to transmit data. These devices, referred to as nodes, have one or more sensors on board which can measure natural phenomena such as temperature, humidity, pressure, sound, light, vibration etc. Sensor nodes talk to each other by using the onboard radio. Typical communication protocols include IEEE 802.11, bluetooth, zigbee etc. Sensor nodes are powered by batteries and hence have a fixed operation life.

WSNs are finding increasing use in a wide spectrum of applications. This is partly because WSNs reduce manual labor and hence cost by automating tasks and partly because WSNs can provide additional capabilities to a system. For example, in traffic monitoring and surveillance applications, a WSN can provide real time event detection and event handling capabilities. In other applications such as environmental monitoring, WSN can collect relevant data for a long period of time over a large deployment area. WSNs are also used in the Industry for various applications like monitoring machine health, pipelines, natural gas leaks, corrosion, goods storage and sales, etc. They are also deployed for tasks like checking the structural integrity of the bridges.

In environmental monitoring applications the sensor network nodes are generally deployed sparsely and are far apart from each other. Therefore, a significant amount of relay nodes (whose only job is to relay data to the base station) may be necessary to form a connected network. Deployment of relay nodes can increase a sensor network's cost significantly. A simple sensor node with a temperature sensor, a radio and which runs on two AA batteries costs around $100. Therefore, in a large scale deployment

where hundreds of relay nodes are required, the cost of sensor network is increased by significantly large amount. Moreover, the cost is also increased due to additional labor required in deployment and maintenance of relay nodes.

Relay nodes also constrain the lifetime of a sensor network. Studies show that communication is a big consumer of energy in sensor nodes [1]. Therefore, in relay nodes energy depletes very fast due to more energy spent in communication. Depending on the topology of the network, a defunct relay node can possibly disconnect a sensor network from its base station. Therefore, researchers are seeking alternative solutions to relay nodes which can reduce a sensor network's deployment cost and can also elongate the life time of a sensor network.

One alternative is to use mobile robots for data collection. Mobile robots can visit each sensor node or an assigned node in a cluster of sensor nodes and collect all the data from the nodes. They can then transmit back the data to the base station. This completely eliminates the need for using relay nodes. Mobile robots can also increase the life time of a sensor network by decreasing the communication energy spent by sensor node. This is because a mobile robot can download data from a sensor node from closer range on a lower radio power.

One might argue that using robots will increase the cost of the system. If we compare a mobile robot's cost with a sensor node's cost, it is significantly high. But as the scale of sensor network increases mobile robot's provide a very cost-effective solution as compared to relay nodes. Also robotic research is already at a stage where fundamental problems of navigation and sensing is solved for many scenarios. In future more sophisticated and robust robotic platform will be available with applicability in wide range of environments. Therefore it is both feasible and cost-effective to use robots for data muling applications.

In this work, we study three fundamental problems which arise in a data muling system. First, in Chapter 3 we start with data collection problem with uniform disk communication model of a sensor. The communication range of each sensor node is represented by a disk of uniform radius. We consider the problem of finding a tour for each of k given robots such that the time to collect data from all then sensor nodes is minimized. We refer to this problem as Data Gathering Problem (DGP). We present an optimal algorithm for the one dimensional version of DGP. For the two dimensional

version, we present a constant factor approximation algorithm. Afterwards, we present field experiments in which an autonomous robotic data mule collects data from sensor nodes deployed over a large area.

During the field experiments we found that a sensor's communication range does not exactly follow the uniform disk model. Further experiments with the sensor's communication range suggested a Two-Ring communication model: an inner ring where robot can download data reliably with a fix download speed and an outer ring where robot can download data with a slower average download speed. The download rate in outer ring is slower because of the packet loss and retransmissions. We study the corresponding data collection problem where goal is to minimize the cost of data collection from sensors with Two-Ring communication model. This problem is referred as Two-ring Tour (TRT)problem. We present a constant factor approximation algorithm for the general case. The algorithm uses a polynomial time approximation scheme as a subroutine. Though the scheme has polynomial running time, its running time is impractically large. It is also complex to implement. Therefore we study special cases of the TRT problem and present efficient algorithms for them.

In the field experiments, we also observed that the robot was running out of energy very quickly. For a data muling system to be useful it is necessary that the robots can operate for extended periods of time. Therefore we started looking into solar energy harvesting for robotic navigation. In Chapter 5, we present an experimental model of energy consumption and harvesting as a function of environmental parameters. We use this model to study the problem of finding an energy efficient trajectory for robot to go from a source point to a destination point such that it collects maximum energy within a given time budget. For this problem we present an optimal algorithm based on dynamic programming.

In the next chapter we present relevant research work in the fields of data muling, routing problems and the problem of energy harvesting using mobile entities.

Chapter 2

Related Work

Research on Wireless Sensor Networks (WSNs) has been very active in the last two decades with researchers focusing on issues such as deployments with coverage guarantees [2], development of energy-efficient communication protocols to improve network lifetime [3], monitoring of humidity levels to determine vineyard irrigation levels [4]. Use of mobility for carrying data in WSN has also received significant attention in recent years. However, most of the existing literature focuses on uncontrolled mobility (such as the mobility of humans). Research on use of controllable entities (such as robots with wireless communication capabilities) for data muling has started to pickup. We start with an overview of related work on data muling.

In [5] a data muling system is presented which uses uncontrolled entities (such as animals, humans with wireless devices) for carrying data. The authors present a three tier sensor network architecture. The bottom layer consists of a sparse sensor network. In the middle layer there are mobile entities such as vehicles and humans which carry the data from bottom layer to the access points in the top level. These ideas were also implemented in real systems in ZebraNet [6] and Smart-tag [7] projects. No assumption is made on the mobility model of the uncontrolled entities.

In the following work, the mobile agents are still uncontrolled however the mobility model is known. In [8], the authors explore the use of observed mobile agents in reducing the energy consumption in sensor networks. In [9], data mules' trajectories are parallel line segments and the goal is to find the balanced assignment of sensors so that the tour times of data mules is minimized. In [10], the authors explore the transmission

scheduling i.e. schedule to wake up a node and transmit to the data mule. The solution is presented under the assumption that the mule follows a random-waypoint mobility pattern. An adaptive data transfer protocol which minimizes the time interval for a single sensor to offload the data is presented in [11]. This protocol considers the packet loss rate due to the distance between sensor and the data mule at the time of offload. A simple protocol is presented in [12] in which nodes send periodic beacons with low duty cycle and turns on their radio when the connection with a data mule is satisfied. A recent review on the state of the art in exploiting sink mobility can be found in [13, 14, 15, 16]

Research on using controllable agents as data mules is very recent. A variant of the Data Gathering Problem (recall that in the Data Gathering Problem problem the goal is to find data collection tours for each of the k given robots such that the time to collect data from all the sensors is minimized) is studied as a scheduling problem in [17]. Authors studies the control of the robot's velocity along a fixed path to improve transmission quality. They do not address on how to compute such path. Similarly, the work in [18] studies the speed control problem for a data mule when it is downloading data from a sensor while traveling within its communication range. Authors present a heuristic which computes the speed change along the mule's path and a schedule for downloading the data from the sensors. In [19], the authors seek a path for each robot such that all paths are disjoint, all sensors are covered and all paths are of equal length. They investigate the performance of their algorithms in simulation. Authors in [20] present a heuristic for the multi-robot boundary coverage problem. In a boundary coverage problem we are given k robots and n convex, two-dimensional objects. The goal is to find a tour for each robot such that all points on the boundary of each object are inspected and the inspection load is balanced. None of the above mention work gives any theoretical guarantees on the performance of their algorithms.

In some research work the Data Gathering Problem is also formulated as TSP instances. In [21], authors present heuristics to schedule visits of a mobile agent to collect data from cluster heads. The heuristics focus on data latency and data aggregation rate of clusters. However in our work we focus on the travel time of the mobile agents and present algorithms with theoretical performance guarantees. [22] present a heuristic approach for minimizing path length in data muling. Authors consider spatially separated WSNs which are to be connected by a data mule. The objective is to find a path for

mule which visits one sensor each in every WSN. Their heuristic creates a path from the convex hull of the set of sensors, chooses one sensor from each WSN, and modifies the path to add the sensors from WSNs not covered by the convex hull. In [23], the authors formulate the problem of collecting sensor data using a single robot as an instance of the TSP with neighborhoods (TSPN) problem. They do not address the time spent in downloading data. None of the above work has been implemented and tested on a real system.

Implementations of data mule systems are presented in [24, 25]. In [24], the authors 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 a good view-angle to start communication. The authors use a TSP algorithm to compute the trajectory of the robot. However since GPS localization is not available under the water, the vehicle has to navigate under high uncertainty and periodically surface to get GPS fix. This makes the designing global routing algorithms challenging. The authors propose spiral movement for the robot 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. Tekdas et al [25] implemented a system which uses TSP and k-TSP algorithms to find efficient strategies for multiple data mules. However they did not incorporate the communication range of the sensors as well as the download times.

The Traveling Salesperson Problem (TSP) is a special case of DGP. It is a fundamental combinatorial optimization problem and have books dedicated to it [26]. Even though TSP in its general form is inapproximable, the metric version admits a constant factor approximation [27]. The Euclidean version in any fixed dimension admits a Polynomial Time Approximation Scheme (PTAS) [28, 29]. A generalization of TSP which received significant recent attention is TSP with Neighborhoods (TSPN) [30, 31]. In a TSPN instance, instead of n points we are given n neighborhoods (a neighborhood is a bounded region). The goal is to find a minimum cost tour which visits at least one point in each neighborhood.

Arkin and Hassin [30] introduced TSPN and presented a $(3\sqrt{2}+1)p$ approximation algorithm for the case when neighborhoods are equal length parallel segments (Here, p is approximation ratio for TSP). For neighborhoods which are translates of a convex

region, they gave a $(\sqrt{7^2+3^2}+1)p$ approximation algorithm.

The version of TSPN where the neighborhoods are uniform disks has many applications. Dumitrescu and Mitchell presented constant factor approximation algorithms for TSPN with uniform disks [31]. They show that a TSP tour that visits the centers of the disks is an $O(1)$-approximation to the TSPN tour.

In more recent work, Mitchell showed that when the disks are disjoint, TSPN admits a PTAS. The result generalizes to other *fat* regions on the plane [32] and can be modified to yield a PTAS for k-TSPN algorithm where the goal is to find the shortest tour which visits at least k disks out of n given disks. Elbassioni et al. presented a constant factor approximation algorithm for intersecting fat convex objects of comparable diameters where the tour is restricted to visit each object only at a finite set of specified points [33].

Energy harvesting with static nodes is a widely studied problem but using mobile entities for energy harvesting is still in nascent stage. There is very limited literature which actively uses mobile agents to harvest energy from environment. [34] studies the problem of using mobile nodes as energy harvesters in a sensor network. These nodes try to keep themselves charged and move to areas where energy is requested to transfer it. Authors compute the number of mobile entities required to guarantee network longevity. In this paper the sole purpose of mobile entities is to harvest energy and goal is to achieve network longevity. We study the problem of harvesting energy while performing the data muling task. Our goal is to compute energy efficient paths for robots. [35] presents a planetary exploration system which uses sun-synchrony to increase operational life. A module called TEMPEST generates sun-synchronous path which is aware of the terrain and position of sun. It re-plans the path based on new information from the sensors. For our problem we needed a solution which generates optimal energy harvesting path while performing the data muling task in a given time budget. In [36] authors studied the problem of finding optimal velocity profile of the robot for a given trajectory such that the energy spent in travelling is minimized. They consider the energy consumed by motors at different speed and acceleration and construct a velocity profile for the trajectory which reduces this energy consumption. While in this work we compute the trajectory where maximum energy harvesting can be done.

Chapter 3

Data gathering problem with uniform disk communication model

3.1 Data Gathering Problem (DGP)[1]

A crucial problem that arises in data muling applications is the problem of planning the routes of data mules: given locations of n sensors, compute the routes of k robots so that the time to download the data from all sensors is minimized. In this chapter, we study this path planning problem which we refer to as the *Data Gathering Problem (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 [26]. 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 [38]. Even though DGP resembles k-TSP, a closer look reveals important differences.

For example, consider the scenario illustrated in Figure 3.1 where the optimal TSP tours for two robots are shown. Two robots are charged with collecting data from the

[1] **A preliminary version of this work appeared in [37]**

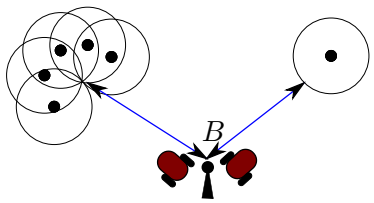

Figure 3.1: k-TSP tour for two robots.

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. This figure shows optimum TSP tours for the two robots which minimize the maximum travel distance by any robot. This solution is not appropriate for data gathering because the robot assigned to the left group would spend significantly more time downloading the data from the sensors. Note that the objective of TSP is to minimize the travel distance. However, in data gathering, downloading the data takes time. If we use the TSP solution for DGP, the robot on the left can spend significantly more time to download the data from all assigned sensors. In fact, we can make the TSP solution arbitrarily bad by increasing the number of nodes in the left cluster.

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 3.1, 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[2] to download the data. This variant of TSP is called TSP with Neighborhoods (TSPN). In the geometric version of TSPN, we are given n disks. The objective is to compute the shortest tour that visits at least one point in each disk.

A instance of DGP consists of the coordinates of n sensors and a base station b. Each of the k robots is charged with downloading data from sensors assigned to it and

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

uploading it to the base station. We define the *coverage of a sensor* as transferring the sensor's data to the base station by a data mule.

We make the following assumptions:

- The sensors are identical. The amount of data to be downloaded from each sensor is the same. Further, the transmission ranges and data sending rates of all 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. We also assume that transmission ranges of all the sensors are obstacle free and a robot can access any point in these transmission ranges.

- The data is downloaded by k identical robots which have wireless communication capabilities and can travel at a constant speed of ν. In this work, we do not consider higher order constraints such as acceleration. Also, without loss of generality, we will use $\nu = 1$. 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. We assume that the communication disk of each sensor is free of obstacles (so that the robot can traverse the boundary of each disk). The robots stop to download data.

We present algorithms for DGP in the succeeding sections under these assumptions, and validate them in field experiments later on.

3.2 Algorithms for DGP

3.2.1 The 1D Data Gathering Problem

In this section, we study the 1D version of the DGP 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 one end point of the curve X; i.e. at $x = 0$ where x is the parametrization of the underlying curve.

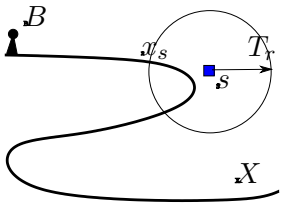

Figure 3.2: A 1D instance of the DGP. Robots can travel only on curve X. x_s represents the download location of sensor s. To download data from s, a robot has to stop at x_s.

For each sensor s, compute the intersection of the communication disk (centered at s with radius T_r) with the curve X. Among all the points of the intersection, let x_s be the point closest to base station. We will choose x_s as the *download location* of 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 k robots.

Let U be the set of robots. Now consider a solution to the 1D DGP. For any two robots $u, v \in U$, let $S(u) = \{u_1, u_2, \ldots, u_k\}$ and $S(v) = \{v_1, v_2, \ldots, v_l\}$ be the sets of sensors assigned to u and v respectively in the solution (we overload the notation and use u_i to refer both a sensor and its download location). $S(u)$ and $S(v)$ are ordered and labeled such that $u_1 \leq u_2 \leq \ldots \leq u_k$ and $v_1 \leq v_2 \leq \ldots \leq v_l$. While ordering download locations, we break ties arbitrarily. We say that $S(u)$ and $S(v)$ are *non-overlapping* if $u_k \leq v_1$ or $v_l \leq u_1$.

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

Lemma 1. *There exists an optimal solution to cover n sensors with k robots in which $\forall u, v \in U$, $S(u)$ and $S(v)$ are non-overlapping.*

Proof. Among all optimal solutions, consider the optimal solution OPT with the largest number of non-overlapping pairs of robots. We claim that there is no overlapping pair in OPT. Suppose, to the contrary, that there is a pair of robots whose sensors are overlapping. We consider two cases. It the first case, there exist two nodes $u, v \in U$ such that $S(u)$ and $S(v)$ are overlapping partially (Figure 3.3 (a)). If this is not the case, then the overlap must be a complete overlap (Figure 3.3 (b)). We show that in

both cases, the overlap can be removed without introducing additional overlaps. This will contradict with the minimality of the number of overlapping pairs in OPT. Without loss of generality, we can assume that $u_1 \leq v_1$.

Case (a): There is a partial overlap of $S(u)$ and $S(v)$. Let the time to cover $S(u)$ (resp. $S(v)$) by robot u (resp. v) be $T(u)$ (resp. $T(v)$). Then

$$T(u) = 2u_k + kT_d \quad \text{and} \quad T(v) = 2v_l + lT_d.$$

In this case, we can reassign sensor $u_k \in S(u)$ to robot v and sensor $v_1 \in S(v)$ to robot u. This will give us new sets of sensors $S'(u) = S(u) - \{u_k\} + \{v_1\}$ and $S'(v) = S(v) - \{v_1\} + \{u_k\}$ that are assigned to u and v. Let $u'_k \in S'(u)$ be the sensor which is farthest from the base station. Then the new coverage times will be:

$$T'(u) = 2u'_k + kT_d \quad \text{and} \quad T'(v) = 2v_l + lT_d$$

Since $u'_k \leq u_k$, we have $T'(u) \leq T(u)$. Also $T'(v) = T(v)$. This reassignment operation does not increase the coverage cost of any of the two robots. We can continue reassigning in the similar way until $u'_k \leq v_1$. This gives us a new assignment for the robots u and v where $u'_k \leq v_1$, or $S'(u)$ and $S'(v)$ are non-overlapping. Note that, since the segments are only getting smaller, no additional overlaps are introduced. Therefore, we reduced the number of overlaps in OPT by one which contradicts the minimality of OPT in terms of the number of non-overlapping pairs.

Case (b): There are no partial overlaps (that is, case (a) is not true). In this case, there must be two sensors u and v such that $S(u)$ completely overlaps $S(v)$ (Figure 3.3(b)). Among all the segments (sensors assigned to a single robot) contained in $S(u)$, let $S(v)$ be the one with the leftmost left-end.

We can reassign sensor $u_1 \in S(u)$ to robot v and sensor $v_1 \in S(v)$ to robot u. This does not increase the cost of coverage for any of the robots u and v. Note that this operation does not increase the number of overlaps because there are no partial overlaps and v is the leftmost among all segments contained in u. After this reassignment, this case becomes similar to case (a). Using the same argument, this means that the number of overlaps can be reduced – a contradiction. □

Lemma 1 sheds light on the structure of sensor assignments in an optimal solution.

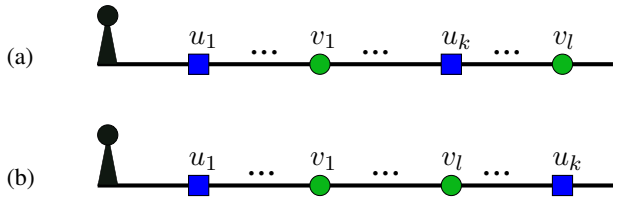

Figure 3.3: Two primary ways in which $S(u)$ and $S(v)$ can be overlapping: (a) partial overlap (b) complete overlap.

We now present a dynamic programming algorithm to exploit this structure and to gather the data from n sensors using k robots in an optimal fashion.

Let's order and label sensors from s_1 to s_n with increasing distance of their download locations from the base station (again, s_i refers to both the i^{th} sensor and its download location). Let $cost(m, l)$ be the cost to cover sensors s_m to s_l, $m \leq l$ by a robot. Therefore $cost(m, l) = 2s_l + (l - m + 1)T_d$. We create an $n \times k$ table T. Each entry $T[i, j]$ represents the optimal cost to cover s_1 to s_i sensors by j robots. The entries of the table are computed using the following recurrence equation:

$$T[i,j] = \begin{cases} T[i,i] & \text{if } j > i \\ 2s_i + iT_d & \text{if } j = 1 \\ \min\limits_{1 \leq m < i} (T[m, j-1] + cost(m+1, i)) & \text{otherwise} \end{cases} \tag{3.1}$$

Lemma 2. *When all entries are computed, $T[i, j]$ stores 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, $T[i, 1] = 2s_i + iT_d \; \forall i \in \{1, n\}$. 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.

Inductive 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 $T[l, j] = \min\limits_{1 \leq m < l} (T[m, j-1] + cost(m+1, l))$. Since $T[m, j-1]$ is the minimum coverage time for m sensors with $j-1$ robots and the value of $T[l, j]$ is set to minimum of all $l-1$ possible values of m, $T[l, j]$ is minimum coverage time of l sensors with j mules. □

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

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

3.2.2 The 2D Data Gathering Problem

In 2D version of DGP, the robot is not restricted to a curve. Instead, it can take any path on the plane. The 2D version of DGP is NP-Hard because it contains the Euclidean TSP problem as a special case. To see this, take a DGP instance where the number of robots is 1, the download time is an arbitrary constant and the transmission range is 0. In this instance minimizing the tour time is equivalent to minimizing the tour length, and the tour visits each sensor point. It is equivalent to an Euclidean TSP instance which is NP-Hard.

In this section, we present an approximation algorithm for 2D DGP. Our algorithm uses tools and techniques developed for the following variants of TSP: TSP with neighborhoods (TSPN) and k-TSP. After an overview of these two algorithms, we show how they can be combined to obtain a constant factor approximation algorithm for the 2D DGP problem.

[31] present a constant factor algorithm for TSPN where the neighborhoods are disks of unit radius. The approximation ratio of this algorithm is 11.15. In this work, we refer to this algorithm by *TspnTour*. *TspnTour* first computes a maximal independent set, say I, of non-intersecting disks from the given set of disks. Next a TSP tour, τ_I, of centers of all the disks in I is computed. A TSPN tour is formed from τ_I as following. The TSPN tour starts from the intersection of the boundary of an arbitrary disk in I and τ_I (a in Fig 3.4). It then follows τ_I in clockwise direction. If the boundary of a disk $D \in I$ is encountered, then D is traversed clockwise along the boundary until the next intersection of τ_I with D is encountered. This continues until the tour returns to starting point. Then τ_I is traversed in similar fashion but in counter clockwise direction until start point is encountered again. This ensures that all the intersecting disks (disks which are not in I) are visited. The final TSPN tour is obtained by combining these

tours.

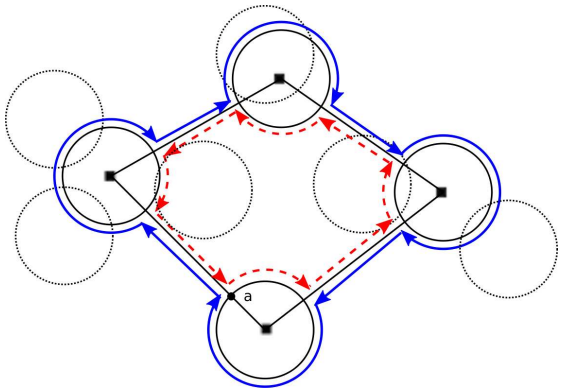

Figure 3.4: A TSPN tour constructed by *TspnTour*. The disks in I are drawn with heavy solid lines. The tour contains two subtours. After visiting a circle, one subtour traverses its boundary clockwise until the next intersection of the disk with the tour. On the way back, the other tour traverses the disk counter-clockwise. Traversing in both clockwise and counter-clockwise direction ensures that all the intersecting disks are covered.

[38] present a constant factor approximation algorithm for k-TSP (*k-SplitTour*). *k-SplitTour* first computes a TSP tour of all the cities. This tour is split into k parts and the end points of each part of the tour is joined to the starting city to form k subtours. The criteria chosen to split the tour ensures that the solution is bounded by a factor of $\alpha + 1 - \frac{1}{k}$, where α is the approximation ratio of the algorithm used for computing the TSP tour.

Now we present our algorithm which uses the above mentioned algorithms to find a constant factor bounded solution to k-DGP problem. We refer to this algorithm by *DGPTour*. The main idea of *DGPTour* is to compute a TSPN tour of sensors and then carefully split it in to k subtours while incorporating the cost of data download. The steps are as follows:

Algorithm DGPTour

1. Find a TSPN tour for n sensors and base station b using *TspnTour* where the neighborhoods are the communication disks of the sensors. Let the tour be $\tau_1 = (b, s_1, s_2, ..., s_n, b)$ and its cost be $|\tau_1|$ (operator $|.|$ signifies the cost of the tour

which includes the time taken to travel plus the time taken to download data). On the tour we fix the download locations for each sensor. The robot stops at a download location to download data from the sensors. For a sensor whose communication disk is in I, its download location is the first intersection of the tour with its communication disk. For a sensor s_i whose communication disk is not in I, there is a sensor s_j which intersects with s_i and is in I. Download location of s_i is fixed to the intersection point of the boundary of s_j and the line joining the centers of s_i and s_j (see Figure 3.5).

2. In this step we find and mark the sensors where we split the tour τ_1. Traverse τ_1 starting from the base station and keep track of the cost of the tour so far. When the tour cost exceeds a particular value from a predefined set of values, the last sensor visited so far is marked for split. Formally, for each $j \in 1, k-1$, among all the sensors along τ_1, find the farthest sensor s_{n_j}, $n_j \in \{1, n\}$ such that the cost (time to travel plus download data) of path from b to s_{n_j} along τ_1 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 farthest download location from the base station.

3. After finding the sensors where the tour will split, we connect the base station to each of the marked sensor and the sensor next to it on the tour τ_1. Then we remove the edge on τ_1 between the marked sensor and the sensor next to it. This gives us k subtours. Or, let s_i^j represents the i^{th} sensor in j^{th} subtour. The j^{th} subtour is represented by $R_j = (b, s_1^j, s_2^j, ..., s_{n_j}^j, b)$ for all $j \in \{1, k\}$, where s_1^j is the node next to $s_{n_{j-1}}^{j-1}$ on the tour τ_1.

In the following we show that the tour obtained by *DGPTour* is a constant factor away from the optimal tour.

Theorem 2. *If* DGPTour *returns τ_k as the maximum cost subtour among all k subtours, and τ_k^* is the maximum cost subtour in the optimal solution for the 2D Data Gathering Problem, then*

$$|\tau_k|/|\tau_k^*| \leq e + 2 - 1/k \tag{3.2}$$

where e is the approximation ratio of the algorithm used to find TSPN tour at step 1 of our algorithm (in our case approximation ratio of TspnTour).

Proof. The proof is similar to the proof of $k-SplitTour$ algorithm for k-TSP presented in [38]. We show how the download times are incorporated.

First we claim that the cost of any subtour R_i produced by *DGPTour* is upper bounded by $(1/k)(|\tau_1| - 2c_{max}) + 2c_{max}$. The cost of the tour τ_1 at sensor s_1^i is greater than $((i-1)/k)(|\tau_1| - 2c_{max}) + c_{max})$ (from step 2) and the cost of the tour at sensor $s_{n_i}^i$ is no more than $((i/k)(|\tau_1| - 2c_{max}) + c_{max})$. Therefore, the cost of the subtour R_i from sensor s_1^i to $s_{n_i}^i$ is at most $(1/k)(|\tau_1| - 2c_{max})$. The cost of connecting the two sensors s_1^i and $s_{n_i}^i$ to the base station can be at most $2c_{max}$. Hence,

$$|\tau_k| = max_{i\in\{1,k\}}(|R_i|) \leq (1/k)(|\tau_1| - 2c_{max}) + 2c_{max} \tag{3.3}$$

Next, let τ_1^* be the optimal tour with 1 robot. We show that the cost of τ_k^* is no less than $\frac{1}{k}$ times the cost of τ_1^*. Or, $\tau_k^* \geq \frac{1}{k}\tau_1^*$. To prove this we create a new tour τ_1' by merging all the k subtours of the optimal solution. We order the subtours and connect the last sensor of each subtour to the first sensor of the next subtour. We delete all but two of the edges joining the first and last sensor of these subtours to the base station. We keep only two such edges, one from the base-station to the first sensor of the first subtour and one from the base station to the last sensor of the last subtour. This gives us a tour τ_1'. For each edge added we deleted two edges and all three of these edges form a triangle. Therefore by the triangle inequality, the new edge will be no more than the sum of the edges deleted. This gives us $|\tau_1'| \leq k|\tau_k^*|$. Also from optimality, $|\tau_1'| \geq |\tau_1^*|$, therefore,

$$\tau_k^* \geq \frac{1}{k}|\tau_1^*| \tag{3.4}$$

Now we will find out the ratio $\frac{|\tau_1|}{|\tau_1^*|}$. Let the cost of TSPN tour at step 1 of *DGPTour* with $T_d = 0$ (only the travel cost) be C and the cost of optimal TSPN tour with $T_d = 0$ be C^*. We have $|\tau_1| = C + nT_d$. Also $|\tau_1^*| \geq \max(C^*, nT_d)$. Therefore,

$$\begin{aligned} \frac{|\tau_1|}{|\tau_1^*|} &= \frac{C + nT_d}{|\tau_1^*|} \\ &= \frac{C}{|\tau_1^*|} + \frac{nT_d}{|\tau_1^*|} \\ &\leq \frac{C}{|C^*|} + 1 = e + 1 \end{aligned} \tag{3.5}$$

where e is the approximation ratio of the algorithm used to find the TSPN tour.

From Equation 3.3, Equation 3.4, Equation 3.5 and the fact that $c_{max} \leq \frac{1}{2}|\tau_k^*|$, we get

$$\begin{aligned} |\tau_k|/|\tau_k^*| &\leq \frac{(1/k)(|\tau_1| - 2c_{max}) + 2c_{max}}{|\tau_k^*|} \\ &\leq \frac{(1/k)|\tau_1| - (1/k)|\tau_k^*| + |\tau_k^*|}{|\tau_k^*|} \leq \frac{(1/k)|\tau_1|}{|\tau_k^*|} - (1/k) + 1 \\ &\leq \frac{|\tau_1|}{|\tau_1^*|} - (1/k) + 1 \leq e + 2 - 1/k. \end{aligned} \tag{3.6}$$

□

Figure 3.5 shows an instance where we use *DGPTour* to divide the tour into two subtours. By fixing the download locations for each sensor, computing the cost of a part of the tour becomes straightforward. This helps in splitting the tour in Step 2 of our algorithm.

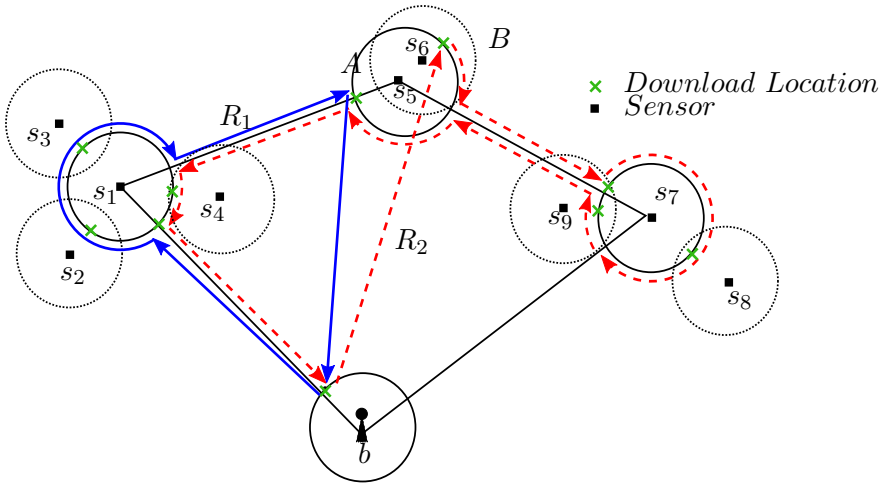

Figure 3.5: Division of a TSPN tour into 2 subtours using *DGPTour* algorithm. Note that the second subtour (dashed lines) visits some sensors which were already visited by the first subtour (solid lines). This is because the original TSPN tour (computed by *TSPNTour* algorithm) visits the disks in both clockwise and anticlockwise directions to ensure that all the intersecting disks are visited. In this figure, the path from s_8 to base station and back to s_9 is not shown to increase clarity.

3.2.3 Data Gathering Problem for Sparse Sensor Networks

In most data muling applications like environmental monitoring the sensors are deployed far apart. In such scenarios *TspnTour* algorithm will have redundant parts because it traverses each edge joining the disks in I (the maximal set of non-intersecting disks) twice. To deal with such scenarios we present an improvement over *TspnTour* algorithm which saves travel cost by not visiting any edge more than once. We refer to the improved algorithm by *SparseTspnTour*. We also formalize the notion of sparsity and provide the condition when *SparseTspnTour* should be used instead of *TspnTour*.

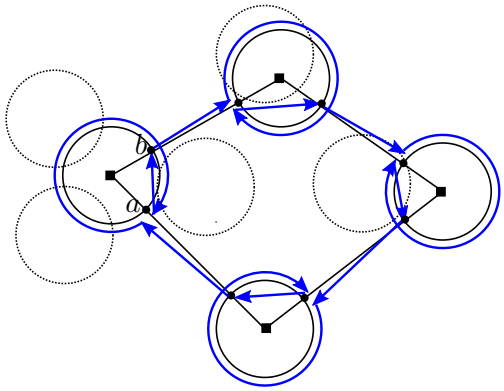

Figure 3.6: This figure shows a TSPN tour computed by *SparseTspnTour*. It covers the boundary of each disk in I (the maximal set of non-intersecting disks) completely after visiting it once. This approach improves the coverage time when the communication disks are far apart.

Overview of the Algorithm SparseTspnTour: First, we compute a tour τ_I of disks in I similarly as in *TspnTour*. We extend it to visit all the disks. To construct the tour start from the point of intersection of the boundary of an arbitrary disk in I and τ_I. Traverse along τ_I until the point of intersection, a, of the boundary of a disk belonging to I 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 τ_I and this disk (Figure 3.6). From b continue along τ_I in similar fashion until the point from where we started is reached.

Let the cost of τ_I in terms of distance be C_I and the number of disks in I be m. Comparing with *SparseTspnTour*, *TspnTour* covers an extra distance of $1/2(C_I -$

$2\pi m T_r$), since it traverses the tour twice. On the other hand, *SparseTspnTour* covers an extra distance of up to $2mT_r$ (along the diameter of the disk to get back on the tour). Therefore, we can use *SparseTspnTour* whenever $2mT_r \leq 1/2(C_I - 2\pi m T_r)$. This gives the condition

$$T_r \leq \frac{C_I}{2m(2+\pi)} \tag{3.7}$$

When the condition in Equation 3.7 is satisfied *SparseTspnTour* performs better than *TspnTour*. The difference in the cost is at least $1/2(C_I - 2\pi m T_r)$. This improvement can be significant in sparse sensor networks like environmental monitoring networks where clusters of sensors are sparsely deployed over large areas. We show in simulations that *DGPTour* based on *SparseTspnTour* performs better than the *DGPTour* based on *TspnTour*. For the field experiments (Section 3.5) we use *DGPTour* algorithm based on *SparseTspnTour*.

3.3 Opportunistic DGP Algorithm

In this section, we present a modification of the DGP algorithm which alleviates some of the challenges faced when executing it on a real system. In particular, there are two major challenges: First, the sensing and actuation errors make it difficult to precisely follow the line segments output by the DGP algorithm. When the robot steers off the computed path, it can be within the communication range of a sensor other than the next sensor it is headed toward. Second, the disk model used for communication is not always accurate. One can choose the disk radius conservatively so that the expected signal quality inside the disk is good. However, occasionally it is possible to receive a good signal outside the disk. We now present a modification of the DGP algorithm which opportunistically downloads from all sensors within the robots communication range (Algorithm 1).

Let $S = \{s_1, \ldots s_n\}$ be the ordered list of sensors assigned to a robot u after the execution of the DGP algorithm. We assume s_1 be the base station so that robot starts its tour from the base in each tour. Let s_i be the next sensor to visit in S. When running the opportunistic algorithm, the robot polls the sensors after s_i and checks if they are within the communication range. Most WSN systems include low-level support for scanning the communication channel. In our implementation, we utilize this capability

of TelosB motes. If the robot hears from any sensor s_j such that $j \geq i$, it stops and collects data from s_j. The robot removes s_j from S, and continues toward s_i. After visiting the last node in S, the robot returns to s_1 ignoring all sensors heard in the mean time because their data has been downloaded recently.

Algorithm 1 *OPPORTUNISTIC_DGP_ALGORITHM*

```
1: Let S be the set of sensors assigned to robot u by DGP algorithm
2: while S ≠ ∅ do
3:    Let s_i be the next sensor to visit
4:    if u hears a good signal from s_i or close enough to s_i then
5:       Stop and collect data from s_i and set S ← S/s_i
6:    else if u hears a good signal from a sensor s_j such that j >= i then
7:       Stop and collect data from s_j.
8:       Set S ← S/s_j and remove the visit of s_j from the DGP tour
9:    else
10:      Continue the tour returned by DGP algorithm
11: Visit s_1 for the next tour then Goto 1
```

3.4 Simulation Experiments

In this section, we further study DGP with simulations. We performed three simulation experiments. In the first simulation, we investigated the utility of increasing the number of robots while keeping the number of sensors fixed. In Figure 3.7, we plot the coverage time (travel time plus data download time) as a function of k, the number of robots. For this experiment we placed 100 sensors uniformly at random in a 600×600 environment. The communication radius T_r was chosen to be 30. As the figure shows, there is a steep decrease in the cost as the number of robots increase initially. But when the number of robots start approaching the number of independent disks (38 in this case), the decrement in the travel cost is not significant.

In the second simulation, we compared the *DGPTour* based on *TspnTour* to *DGPTour* based on *SparseTspnTour*. We placed 30 sensors uniformly inside a rectangle

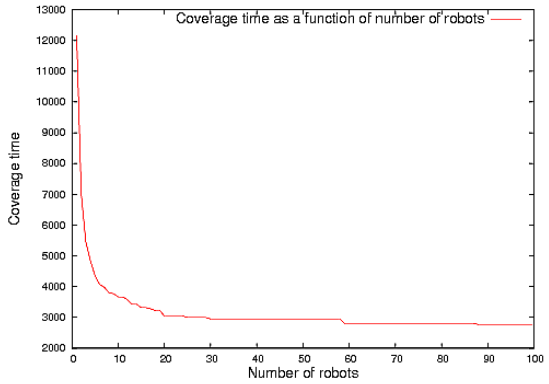

Figure 3.7: Coverage time as a function of robots for 100 sensors deployed uniformly at random in a 600×600 area. Coverage time includes the time taken to visit all the disks and the time taken to download data from them.

of size 600×600. The download time was fixed to 50 units and the transmission radius was set to 30 units. We performed 100 trials. In each trial we computed tours for two robots using the two algorithms. Figure 3.8 shows the computed tours for one such trial. The average tour cost obtained for *DGPTour* based on *SparseTspnTour* was 2487 units with a standard deviation of 153 units. For *DGPTour* based on *TspnTour* the average tour cost computed was 4402 units with a standard deviation of 294 units which is approximately 77% costlier. This is because the total cost of the subtours obtained by *TspnTour* based *DGPTour* is dominated by the cost of traveling the edges between non-intersecting disks twice in the original TSPN tour. Since *SparseTspnTour* based *DGPTour* avoids traversing each edge twice, it is significantly shorter than the TspnTour based algorithm. Therefore, in sparse deployments *SparseTspnTour* is a more effective algorithm.

Finally, we compared the performance of *DGPTour* based on *SparseTspnTour* with the following two alternatives: The first one is the commonly used alternative of a k-TSP tour of centers of the disks (i.e. sensor locations). The second is a variant of SparseTspnTour where we compute a k-TSP tour of the download locations of sensors which are computed in Step 1 of *DGPTour* algorithm. This is a viable heuristic because SparseTspnTour computes a tour of disks in the independent set but for the remaining disks, their visit order is given by the order along the boundary of the disks they

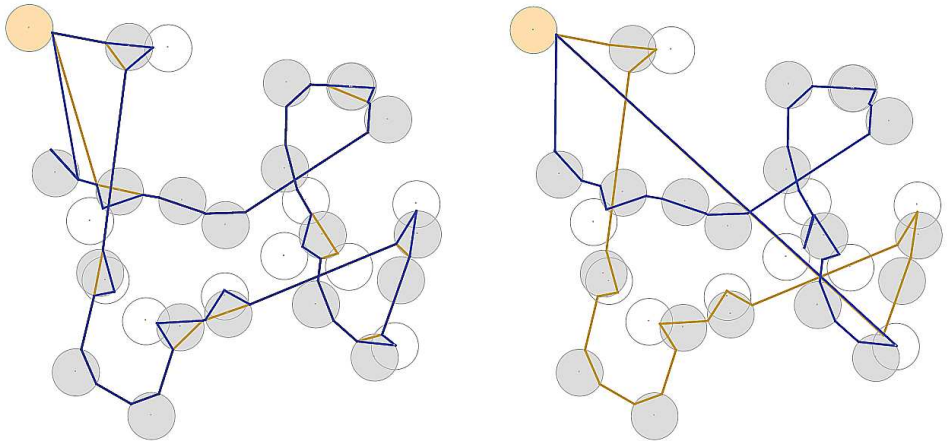

Figure 3.8: An instance of a sparse network where *SparseTspnTour* performs much better than *TspnTour*. The shaded disks show the disks which belongs to I. The base station is the top left disk in yellow (lighter) shade. **Left:** The DGP tour for two robots which is computed using *DGPTour* based on *TspnTour*. **Right:** The DGP tour for two robots computed by using *DGPTour* based on *SparseTspnTour*.

intersect. A natural question is whether it is worth computing a TSP tour over the entire set of points once they are generated by SparseTspnTour.

We performed 100 trials and in each trial we placed 80 sensors uniformly at random inside a rectangular area of size 600×600. Download time for each sensor was set to 50. For 1, 2 and 4 robots we computed tours while increasing the transmission radius from 40 to 80 and then 120. We computed the TSP tour of centers of the disks in the non-intersecting set by using a 1.5-approximation algorithm presented in [27].

The difference of average costs of the tours obtained and the DGP tour is reported in Table 3.1. Since k-TSP tour of centers is constructed by visiting centers of all the disks, it does not take advantage of the neighborhood. Therefore the performance of k-TSP deteriorates as compared to *DGPTour* with increase in transmission range. On increasing the transmission range *DGPTour* performs significantly better. This is because the TSPN tour becomes shorter with increase in transmission range. Figure 3.10, Figure 3.11 and Figure 3.12 show the k-TSP tours of centers and the DGP tours. It is easy to see that the TSPN tour length decreases significantly in Figure 3.12(Right).

Algorithms	**Number of Robots**	**Cost difference with DGPTour as percentage**		
		Transmission Range: 40	Transmission Range: 80	Transmission Range: 120
k-TSP of centers k-TSP of download locations	1 1	-2.70 10.06	-1.12 -3.71	5.46 -5.25
k-TSP of centers k-TSP of download locations	2 2	-1.63 9.83	0.90 -1.99	6.80 -3.62
k-TSP of centers k-TSP of download locations	4 4	-1.09 8.06	1.50 -1.84	6.80 -3.36

Table 3.1: Comparison of DGPTour based on SparseTspnTour with two alternatives. The algorithms are: k-TSP tour of sensor locations and k-TSP tour of the download locations generated by SparseTspnTour. The values given are the percentage difference of the cost of each algorithm to the performance of DGPTour. Positive (resp. negative) difference value means that the tour cost is higher (resp. lower) than the DGP tour cost by that percentage value.

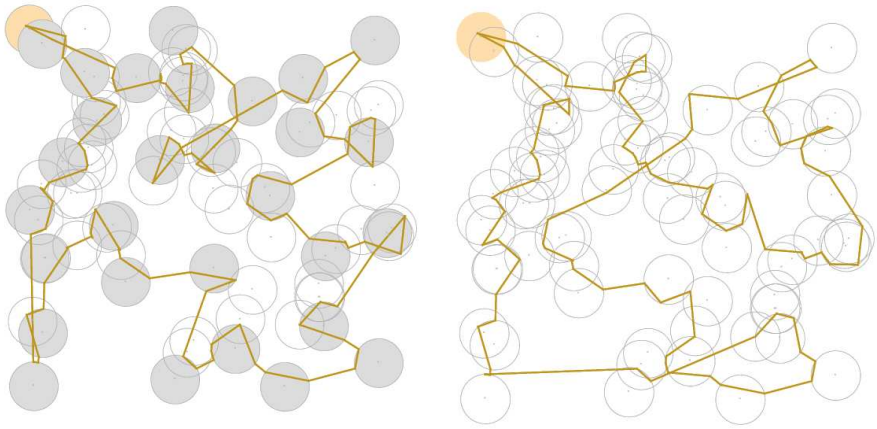

Figure 3.9: An instance where the DGP tour performs better than the k-TSP tour of download locations by 15%. Transmission range of the sensors is 40. The base station is the top left disk in yellow (lighter) shade in both the figures. **Left:** Data gathering with one robot using *DGPTour*. **Right:** Data gathering with one robot using k-TSP tour of download locations.

The cost incurred by the third algorithm (k-TSP of the download locations) is mostly lower than the DGP tour but is occasionally higher. This is surprising because theoretically the optimal k-TSP tour of the download location will always be less than the DGP tour as both the tours visit the same vertex set. In the implementation we use an approximation algorithm for k-TSP [38] to compute the tour. Therefore the tour cost can be at most $(\frac{5}{2} - \frac{1}{k})$ times the optimal solution. In instances where SparseTspnTour is better than this algorithm, it must be that the heuristic is performing worse than the SparseTspnTour. Figure 3.9 shows an instance where the performance of k-TSP of the download locations is worse than the DGP algorithm. In this instance the cost of the k-tsp tour is 15.2% higher than the cost of the DGP tour.

In the next section, we present results from a real deployment of our data mule system.

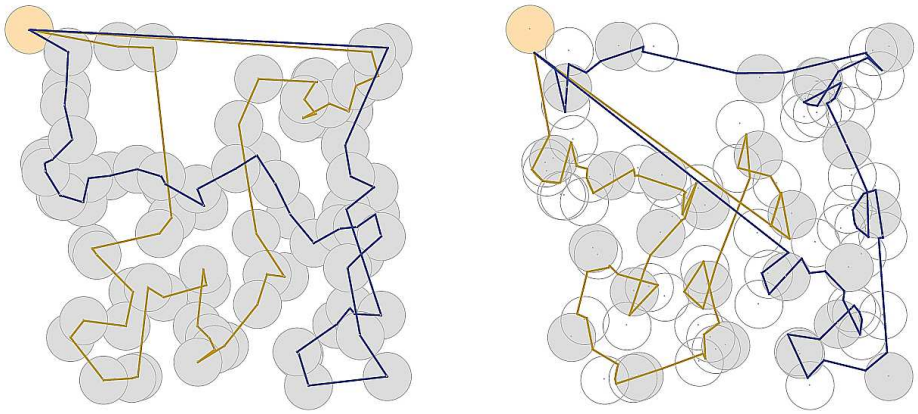

Figure 3.10: Transmission range of the sensors is 40. The base station is the top left disk in yellow (lighter) shade in both the figures. **Left:** Data gathering with two robot using k-TSP. **Right:** Data gathering with two robots using *DGPTour*.

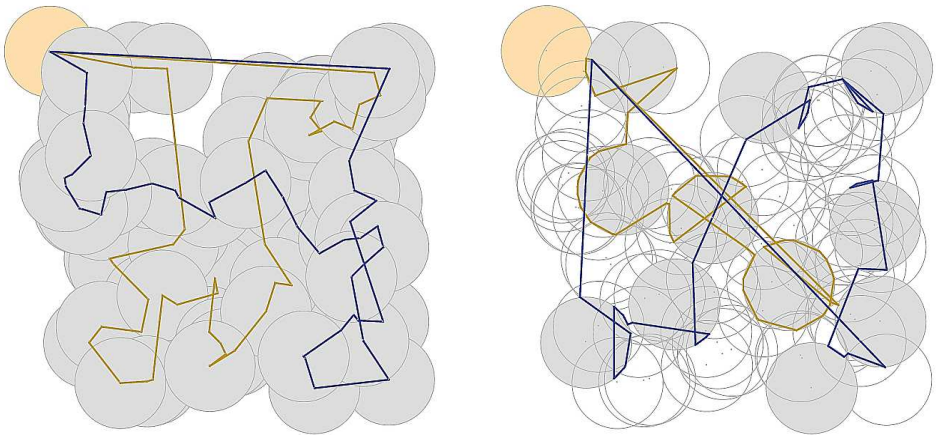

Figure 3.11: Transmission range of the sensors is 80. **Left:** The subtours computed by k-TSP does not change with increase in transmission range. This is because k-TSP does not include the neighborhoods in computing the tours. **Right:** On the other hand the subtours calculated by *DGPTour* shortens in length.

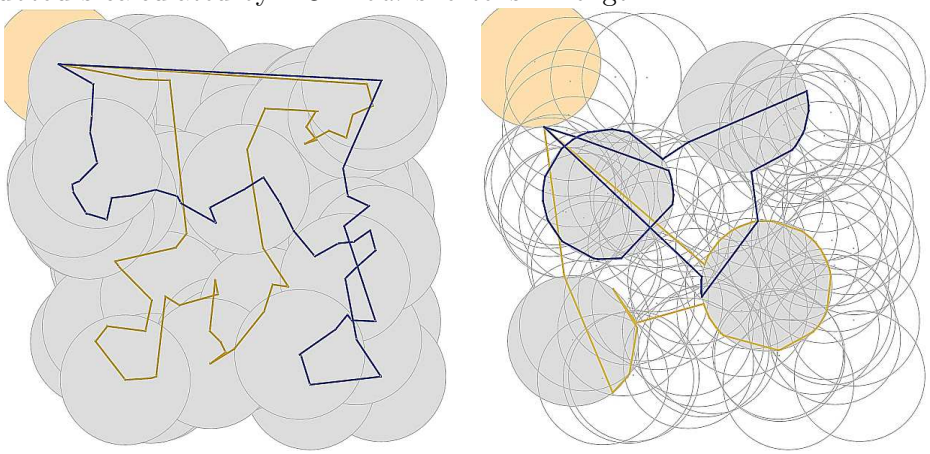

Figure 3.13: **Left:** Cyclops robotic platform developed for data muling. **Right:** A data mule system together with sensors.

3.5 Field Experiments[3]

In order to demonstrate the feasibility and utility of using robots for data collection, we developed an autonomous data muling system. In this section, we first present the system components and details of our implementation. Next, we present results from field experiments.

3.5.1 System Description

In this section we describe the hardware and software components of our system.

Hardware

Since sensor networks are typically composed of inexpensive components, we focused on developing an inexpensive yet robust and effective system so that it can be used commonly e.g. in environmental monitoring. Considering these design choices, we developed an outdoor robotic platform which we call "cyclops" (Figure 3.13).

The entire system is composed of the following equipment and devices:

- Robotic data mule:
  - Radio controlled racing truck base (Tamiya TXT-1)

[3] **In collaboration with Onur Tekdas and this work is reported in [39]**

  - Motor driver (Novak Super Duty XR)
  - Micro controller (Robostix)
  - Laptop computer (Asus Eee PC)
  - Infra Red (IR) sensors (Sharp GP2Y0A02YK)
  - Digital compass (Sparkfun HMC6352)
  - GPS (BU-353)

- Environmental sensors (Crossbow TelosB motes)

The cyclops is based on an RC truck with front servo steering. In order to increase its maneuvering capabilities, we installed a back servo motor which steers in the back wheels in the opposite direction of the frontal wheels. A robostix micro controller is interfaced with the motor driver and steer servo motors to control the robot. For obstacle avoidance we placed four IR sensors in front. One sensor is pointed directly forward whereas two sensors look 45 degrees to the sides. This configuration of IR sensors allows the robot to cover a large range of frontal view. We also placed a fourth sensor looking down to act as a cliff sensor.

We use a compass and a GPS for estimating the robot's pose during navigation (We present the details about localization and navigation in the next section.). Compass and IR sensors are interfaced with a robostix controller. We use a laptop for computing and executing the motion plan. The laptop is interfaced with robostix through serial port. It sends drive commands and receives sensor reading through this serial port. In addition, a GPS device is directly attached to the laptop's serial port. There are three power sources on the robot. The first one is a 12V battery which powers the motors. Robostix controller and the laptop run on dedicated batteries. The battery life of the entire system is approximately two hours indoors. However we observed that this time could drop to half an hour outdoors depending on the environment. Even though this is sufficient for field tests, a longer battery life is desirable for real-life deployments. We are currently investigating the use of solar power to address this issue.

Finally we use TelosB motes as stationary sensor devices. TelosB motes are equipped with a USB interface, an IEEE 802.15.4 radio, a low-power CPU and on-board sensors (temperature, light and humidity) which are powered by two AA batteries. Using low-power sensing, these wireless sensor devices make it feasible to monitor environments

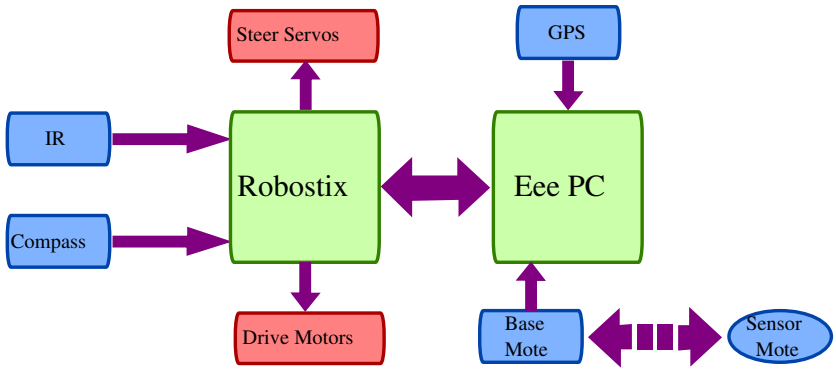

Figure 3.14: System diagram. Robostix controls drive and servo motors, reads data from IR sensors and compass. Eee PC communicates through robostix to control the robot and reads data from GPS. Base mote downloads data from sensor motes and sends the data to the laptop.

for months. We use motes for two purposes. Motes of the first type are called *sensor motes* which are deployed onto the field and collect data from the environment. The motes of the second type are called *base motes* which are attached to the laptops on the robots and used for enabling the communication between sensors and the laptop. A base mote communicates with sensor motes using ZigBee protocol and forwards the messages to the serial port of the laptop and vice versa to enable this communication. A diagram of our overall system is shown in Figure 3.14.

As discussed earlier, we made our design choices so as to construct an inexpensive and robust robotic platform for environmental monitoring applications. Cyclops has light aluminum ladder frame and multi-link suspension to make it possible to navigate in rough domains with high speed (max. 6 kmph) while carrying all the payload (about 7 kg including the base platform). Moreover including all the equipment and devices, the cost of the entire system (not including sensor motes) is about $1,000.

Navigation Algorithms and Software

The software portion of our system consists of two components. The first component is an embedded program developed for robostix. This component is implemented in C++ and provides an Application Programming Interface (API) for controlling the robot

and reading from compass and IR sensors. The second component is developed for the laptop and used for high level control (navigation and sensory data download). Since we have various sensors which have to be read in parallel, we implemented a multi-threaded system. A separate thread reads and writes from/to each device (compass, GPS, robostix and base mote) and updates its state in the main thread. Since Java provides an advanced threading scheme, we used Java to implement the high level application. We also implemented a server/client application and a GUI to visualize the robot's state from a remote computer.

Motes are programmed in nesC. Sensing motes send simple beacon messages, while the base mote receives these messages and filters them according to their mote ids. Messages coming from the target mote are processed in the base mote. The base computes the link quality indicator (LQI) values of the packets received and sends them as serial packets to the Java program running on the laptop. The Java program saves the timestamps (start and end time of the download) and packets received.

The DGP algorithm in Section 3.2.3 yields way-points to be visited along with the ids of sensors whose data must be downloaded at each way-point. We now describe the algorithm for navigating between these way-points.

The robot uses only compass and GPS for pose estimation. Since neither sensor provides accurate information, the robot has to deal with large uncertainties during navigation. At the beginning of its tour, the robot executes an initialization routine to ensure that it receives the most accurate readings from these devices.

The compass requires an initial calibration step to adjust its values according to the earth's magnetic field and external electro-magnetic fields. For that purpose, robot loops around a circle twice at the beginning and calibrates its compass. Afterward the robot uses only compass measurements to determine its heading.

It is well-known that GPS performance changes according to external factors (e.g. environmental effects, and the number and configuration of satellites). To improve its accuracy, we enabled Wide Area Augmentation System (WAAS) feature of our GPS device which corrects for GPS signal errors caused by ionospheric disturbances, timing, and satellite orbit errors. Moreover, we determined that GPS accuracy is higher when robot is moving compared to when it is stationary. Hence, initially to get a better

estimate, the robot moves on a straight line until the variance of the heading measurements drops below a threshold. After good GPS measurements are obtained, the robot computes its desired heading by using its current location and the target location (next way-point). The robot uses its compass to move in the desired direction. However, due to errors, sometimes it goes off of the desired trajectory. The opportunistic algorithm presented in Section 3.3 overcomes some of the inefficiencies associated with this type of error. When the error goes beyond a threshold, the robot re-computes its trajectory to the next node. In experiments described next, the uncertainty of GPS measurements was about 3 meters.

3.5.2 Experiments

In this section we present field experiments conducted by the system discussed above. These experiments are designed for two reasons. First we aim to show the practical feasibility of using an inexpensive robot as a data mule for collecting data from sensors. Second we aim to show how our algorithm can be improved in practice under navigation and communication uncertainties. For this purpose we conducted two sets of experiments. In the first experiment we showed the feasibility of our system using DGP algorithm presented in Section 3.2.3. Using observations from the first experiment, we extended our algorithm by using the opportunistic approach presented in Section 3.3. Since we have only one robot, our experiments do not involve multiple robots. However, since the DGP algorithm partitions the sensors among the robots and robots do not communicate during the data collection, in a single robot is sufficient to accomplish our goals.

All experiments were conducted in East River Flats near the Mississippi River in Minnesota. The ground surface of this field is mainly covered with long grass which makes it a challenging environment for navigation. See Figure 3.15. We placed seven sensors ($S = \{S0, \ldots, S6\}$) in a rectangular area of approximate size of 170m$\times$70m. Since a base and a sensor are treated as identical disks, we did not deploy a specific base. Instead, we treat the last sensor as the base station. Sensors' communication range was set to 6m. In our deployment, the communication disks of sensors S0, S1 and S2 overlap as well as the communication disks of sensors S4 and S5 which were placed at the top of a hill.

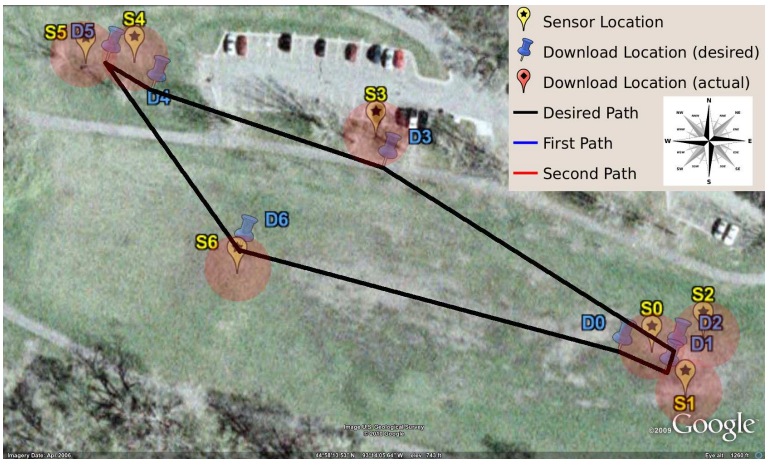

Figure 3.15: Field deployment in East River Flats: $\{S0, \ldots, S6\}$ are sensor locations. $\{D0, \ldots, D6\}$ are the respective download locations computed by our DGP algorithm. Red (shaded) disks show the communication disks and black tour shows the ideal TSPN tour.

We computed a tour for this deployment using the *SparseTspnTour* algorithm presented in Section 3.2.3. *SparseTspnTour* chose $S0$, $S3$, $S4$ and $S6$ as sensors with non-overlapping communication disks in I. For each overlapping sensor we compute a download location. The ordered tour is given by: $\tau = \{D0, D1, D2, D3, D4, D5, D6\}$. The robot was programmed to download twenty packets of size 32 bytes from each sensor where each sensor was programmed to send packets as beacons with time interval 250 milliseconds.

In the first experiment we made the robot execute the tour τ. Due to GPS uncertainties we programmed the robot to stop if its distance to destination point Di is closer than some threshold. Moreover during our initial experiments we realized that the robot occasionally hears good signal from long distances while moving towards the target sensor due the environmental effects. For example robot was able hear the sensor at the top of the hill ($S4$) from approximately 25m distance. In these cases robot does not have to navigate all the way to the desired download location. Hence we modified our algorithm by adding the following condition: robot collects data from sensor Si if it is close enough to Di or it hears a good signal from Si. Note that this modification does not allow downloading from nodes out of order. It simply changes the download

location if a better signal becomes available early on.

Left and right images in Figure 3.16 show the GPS trace of the robot during the first and the second tours respectively. The actual download locations were marked with the different placemarks and labeled as $\{F0, \ldots, F6\}$. The actual and intended download locations were different due to the uncertainty in GPS readings and the modification regarding the download location. The download, travel and total times are reported in Table 3.2. We also include the total tour time as the sum of download and travel times.

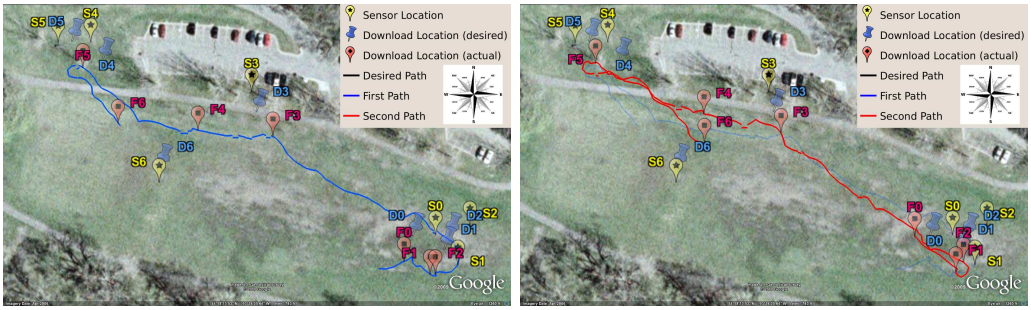

Figure 3.16: Left: First tour by the robot (blue trace). Fi is the actual location from where the robot downloads the data from Si for $i = 0, \ldots, 6$. Right: Two complete tours of the robot where first tour is marked in pale blue and the second tour in red. These images are best viewed in color.

Since the actual starting positions in this experiment and the next one were different, we started the first tour from $S0$. Hence the travel time to reach $S0$ was not counted in the first tour. However in the second tour the travel time from $S6$ to $S0$ was counted as the travel time to reach this sensor. Observe that the download times are consistent with each other except the last sensor in the second tour. This is because, while moving towards $S6$ robot heard a temporary good signal from $S6$ and stopped. However the signal strength dropped during the download which extended the download time.

In the first experiment, we realized that the robot hears from other sensors while moving towards target location. This specially happens when robot is collecting data from overlapping sensors. For example, when robot is moving towards $S1$, sometimes it hears a good signal from $S2$ before $S1$. In this case robot can download data from $S2$ and therefore can eliminate the traveling cost to $S2$. Hence we proposed an opportunistic version of the DGP algorithm which is presented in detail in Section 3.3. In contrast to

	Tour-1		Tour-2	
	Download time	Travel Time	Download Time	TravelTime
S0	4.9		5.3	151.1
S1	5.1	12.2	5.9	26.0
S2	5.1	1.2	4.8	0.9
S3	5.0	100.0	5.1	113.7
S4	4.9	26.7	4.9	33.9
S5	4.8	64.5	4.8	57.3
S6	5.0	41.7	30.0	100.5
Total	34.84	246.2	60.93	483.33
Tour Total	825.3			

Table 3.2: Download and travel times in seconds from the DGP experiment. The first row of Tour-2 includes the time to travel from S6 to S0.

download location modification, this modification allows robot to download from any node that the robot can hear which consequently changes the order of the tour.

Left and Right of Figure 3.17 show the GPS trace from the opportunistic DGP experiment. The download and travel times during this experiment are reported in Table 3.3. In this experiment robot leverages the opportunistic approach to reduce its tour time. The total travel time in the first experiment was 730 seconds whereas in the opportunistic DGP experiment the total travel time was 708 seconds. However it is hard make a reliable comparison between the two strategies since the actual paths and the actual download locations were different. On the other hand we can see the advantage of the opportunistic approach when robot downloads data from $S2$ in the first tour. Since $S0, S1$ and $S2$ were close robot heard a good signal from $S2$ when moving from $S0$ to $S1$ and download the data on the way. Hence it did not spend extra time to visit $S2$ which otherwise it would have spent about 13 seconds to navigate to $D2$ as in the second tour.

Next we compute how much the robot deviates from the ideal solution shown in Figure 3.15. The length of the ideal tour is 550m. The length of the robot's path in the first experiment was 625m. In the second experiment (opportunistic) this length was 629m. Therefore the uncertainties increase the tour length by 13.6% and 14.3%

	Tour-1		Tour-2	
	Download time	Travel Time	Download Time	TravelTime
S0	4.8		5.0	97.5
S1	4.8	46.6	4.7	8.1
S2	5.0	0	5.3	13.4
S3	4.8	113.3	6.3	111.4
S4	5.8	28.5	5.7	35.0
S5	4.9	47.6	8.2	72.5
S6	5.9	42.8	5.6	91.2
Total	36.08	278.75	40.93	429.07
Tour Total	784.82			

Table 3.3: Download and travel times in seconds from the opportunistic DGP experiment. The first row of Tour-2 includes the time to travel from S6 to S0.

respectively. In terms of download times ideal solution would have spent at least 65.8 seconds to download data from all the sensors (assuming 4.7 seconds download time from each sensor). On the other hand in the first and the second experiments the total download times were 95.8 and 77 seconds, respectively. These yield to 45.6% and 17% increase in the download times. In terms of total tour times ideal solution would spend 615.8 seconds to finish two tours assuming a 1 m/sec average speed. The total tour times in the first and the second experiments were 825.3 and 784.8 seconds respectively. Hence the overall deviations from the ideal solution were 34% and 27.4%.

Although the number of experiments conducted were limited, the results show that despite the navigation and communication uncertainties, the performance of the system was not too far from the ideal solution. More importantly, they demonstrate that a data muling system built from inexpensive off-the-shelf components is feasible. In the future we aim to reduce navigational uncertainties by improving our algorithms and hardware capabilities.

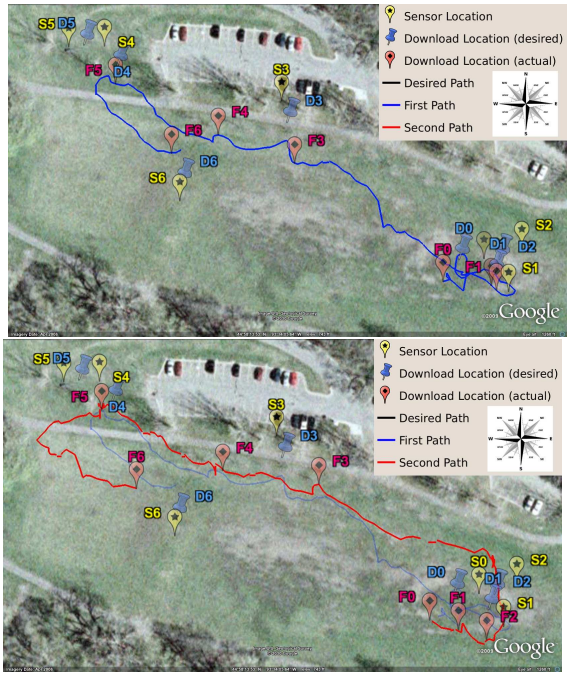

Figure 3.17: Left: First tour from the opportunistic DGP algorithm (blue trace). Fi is the actual location from where the robot downloads the data from Si for $i = 0, \ldots, 6$. Right: Two complete tours of the robot where first tour is marked in pale blue and the second tour in red. These images are best viewed in color.

Chapter 4

Data gathering with two ring communication model[1]

The uniform disk model ignores the fact that download time is a function of the signal strength. In this chapter, we use the *two ring communication model* which addresses the dependency of communication quality on signal strength explicitly. In the two ring communication model, there are two concentric disks centered at the sensor location. Inside the inner disk the communication is reliable thus the download time is shorter. Between the boundaries of the inner and the outer disks, communication is possible however due to increase in packet loss, the download time is longer (see Figure 4.1). Collecting data under *two ring communication model* introduces a new optimization problem which we call *Two-Ring Tour Problem* (TRT). In this version of the problem the tours are no longer optimal TSPN paths. An optimal tour is the one which trades-off between downloading from the outer disk or going further to download from the inner disk with shorter download time.

4.1 The Model and Motivating Examples

Before presenting TRT algorithms, in this section we validate the two ring communication model. We performed experiments using wireless sensors (telosB motes) and studied the dependency of download time on distance. Figure 4.2 shows the average

[1] **In collaboration with Volkan Isler and Onur Tekdas**

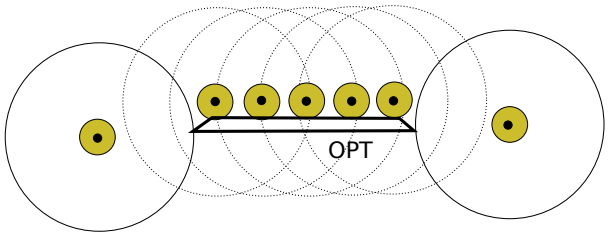

Figure 4.1: A TRT instance. Each sensor has a two ring communication model. If the robot enters the inner disk (shaded region), it downloads data faster than downloading from the outer disk. For this instance the optimal solution visits a mixture of inner and outer disks.

download times as a function of distance from the sensor. When the distance is less than a threshold (18 feet), communication is reliable and the download time is very short. Between 18 and 30 feet due to packet loss download time might be very long. Further from 30 feet threshold either there exists no communication or download time is arbitrarily long. Geometrically this suggests the *two-ring communication model*: For each sensor node s we are given two concentric rings centered at s. The *expected* download times are uniform inside these rings however the *expected* download time inside the inner disk is smaller than the *expected* download time inside the outer ring (see Figure 4.5).

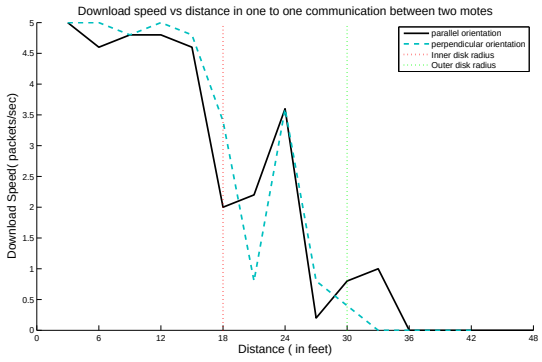

Figure 4.2: Download speed vs distance for communication between two sensor motes. It indicates that the download time can be geometrically modeled as two rings.

We also conducted real experiments with a robot to compare the performance of

various TRT tours. We used Cyclops robotic platform to collect data from four telosB sensors. These sensors were deployed on the corners of a square field of size 70x70 feet (See Figure 4.3). The previous experiment suggests that the inner disk radius is 18 feet and the outer disk radius is 30 feet.

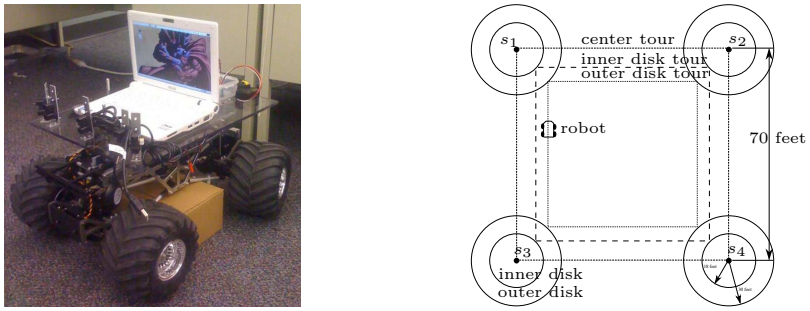

Figure 4.3: **Left:** Cyclops robotic platform **Right:** The setup of the experiment.

We tested three natural strategies and compared their performances. In the first strategy the robot followed a TSP tour which visits the centers. In the second and third strategies, the robot visited the outer and inner disks respectively. In each visit the robot downloaded 100 packets and the time to download varied according to the packet loss.

First we treated the TRT problem as a TSP problem and made the robot visit the sensor locations. In this case the total download time was 8 seconds where as the travel time was 140 seconds (see Figure 4.4 and Table 1). In the second experiment, the robot visited the outer disks which dropped the tour time from 148 seconds to 114 seconds. In this case the download time and the travel time were 50 seconds and 64 seconds respectively. In the last experiment, the robot followed an inner disk tour. The download time drastically decreased from 50 seconds to 9 seconds in this case. However, the travel time increased from 64 seconds to 119 seconds and the overall tour time increased by 14 seconds to 128 seconds.

This simple motivating example shows us that choosing the right download location drastically affects the efficiency of a data muling system. In the subsequent sections, we address the problem of finding efficient TRT tours and present algorithms with proven performance bounds.

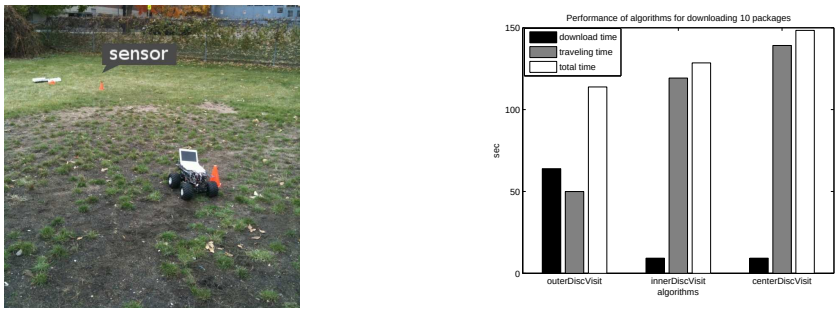

Figure 4.4: **Left:** A snapshot from the experiment. **Right:** Download and travelling time of the following strategies: visit outer disk, visit inner disks and visit disk centers.

	Download Time	Travel Time	Total Time
Visit Disk Centers	8	140	148
Visit Outer Disks	50	64	114
Visit Inner Disks	9	119	128

Table 4.1: Table showing the download and travel times from the three strategies.

4.2 Problem Definition

Let $S = \{s_1, \ldots, s_n\}$ be a given set of the locations of n sensors. For each sensor $s \in S$, define D_{in} as the *inner disk* (i.e. the disk centered at s with radius r_{in}) and D_{out} as the *outer disk* (with radius r_{out}). The download time inside the inner disk is T_{in} and the download time inside $D_{out} - D_{in}$ is T_{out}. Without loss of generality, we scale the distances so that the robot's maximum velocity is one unit.

The objective is to find a tour which visits either the inner disk or the outer disk of each sensor and minimizes the total time taken to travel and download data from all sensors. We refer to this problem as the *Two-Ring Tour Problem (TRT)*. We also assume that the robot stops first and then downloads the data. Observe that the robot can reduce the total time of the tour by at most half if it downloads while travelling. Therefore our results yield approximation algorithms for this model (increased by a factor 2) as well.

The TRT problem is a generalization of the Eucledian TSP problem which is a

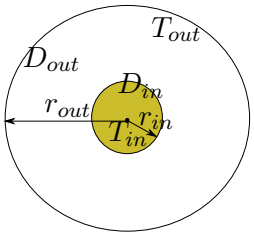

Figure 4.5: The two ring model. Download time is T_{in} in disk D_{in} while it is T_{out} in the region $D_{out} - D_{in}$.

NP-HARD problem. This implies that the there is no algorithm which can solve the TRT problem in polynomial time unless $P = NP$. For such problems, approximation algorithms can be used to get an approximate solution. An approximation algorithm is an algorithm which runs in polynomial time on the size of input and guarantees that the solution is close to the optimal value. For minimization problems like TRT, an $\alpha-$approximation algorithm gives a solution which is at most α times of the optimal value. In Section 4.4 we present approximation algorithms for the general TRT problem.

4.3 Structural Properties

In this section, we present some basic properties of an optimal TRT solution.

Proposition 1. $|C_{in}| \geq |C^*_{out}|$, *where* $|C_{in}|$ *is any tour that visits all inner disks, and* C^*_{out} *is the optimal TSPN tour visiting outer disks.*

Proposition 1 follows from the fact that any tour that visits the inner disks also visits the outer disks. Using Proposition 1 we obtain a lower bound on the cost of the optimal solution to the TRT problem.

Proposition 2. $OPT \geq |C^*_{out}| + nT_{in}$, *where OPT is the cost incurred by the optimal TRT tour.*

The first term on the right side of the inequality in Proposition 2 is the lower bound on the travel time taken by the optimal tour that visits all the sensors, while the second term is the lower bound on the time to download data from all the sensors.

The following theorem yields a lower bound on the optimal solution. We will use this lower bound in some of the approximation algorithms presented in this paper.

Lemma 3. *For any three non-overlapping, equal-size disks on the plane, the length of any path that visits all three disks is lower bounded by αr, where r is the radius of the disks and $\alpha = 0.4786$.*

Proof. Let τ^* be the optimal TSPN solution visiting $n > 2$ disks. Let D_1, D_2 and D_3 be three consecutive disjoint disks on τ^* with centers c_1, c_2 and c_3 respectively. Consider the segment of τ^* which visits these disks. τ^* either touches the boundary of a disk or it crosses the boundary twice. For each disk D_i, we identify a point t_i which is either the touching point or one of the points where τ^* crosses D_i as follows (see Figure 4.6):

- If τ^* touches D_2, pick the touching point as t_2, otherwise pick one of the two points on the boundary of D_2 arbitrarily which is crossed by τ^*

- If τ^* touches D_1, pick the touching point as t_1, otherwise pick the closest crossing point to t_2 as t_1 (similarly choose t_3)

We will present a couple of transformations on the disks such that the length of τ^* will not increase. Afterwards, we will present a lower bound on $|t_1\ t_2| + |t_2\ t_3|$.

In the first transformation, we will replace points t_1 and t_3. In the second transformation, we will move disks D_1 and D_3 in such a way that D_2 touches both disks. Both of these transformations will be done without increasing the total distance. Finally, we will establish the lower bound by optimizing the location of t_2.

Let t_1' be the point that segment $[t_2\ c_1]$ crosses the boundary of D_1. By moving t_1 to t_1', we do not increase the total distance (see Figure 4.6). Same observation can be applied for t_3 and t_3'.

If either $|t_1\ t_2|$ or $|t_2\ t_3|$ is greater than or equal to $0.4786r$, lower bound holds since the total distance is at least as claimed . If both distances are less than $0.4786r$ then we do the following transformation. Without loss of generality, let us assume that D_1 is to the left of D_3 (see left of Figure 4.7). If D_1 touches D_2, we do not move D_1. Otherwise, we rotate D_1 along t_2 in counterclockwise direction until D_1 touches D_2. Similarly, if D_3 does not touch D_2, initially, we rotate it in clockwise direction until it touches D_2. Note that this transformation is only a rotation and does not change the total distance. Middle of Figure 4.7 shows this transformation.

In this transformed version of the problem, we formulate the total distance in terms of parameters $\theta = \angle t_2\ c_2\ c_3'$ and $\beta = \angle c_1'\ c_2\ c_3'$ where c_1' and c_3' are centers of transformed disks D_1 and D_3. Since t_2 is on the boundary of D_2, we can define all possible locations

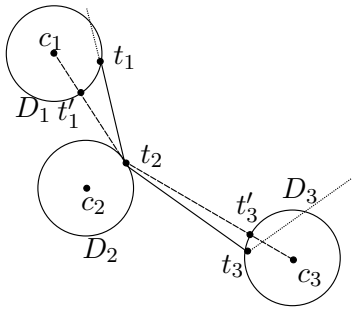

Figure 4.6: Three non-overlapping disks lying on a plane. The part of the optimal TSPN tour t_1, t_2, t_3 which visits disks D_1, D_2 and D_3, respectively. Without increasing the total distance, we can transform t_1 to t'_1 and t_3 to t'_3

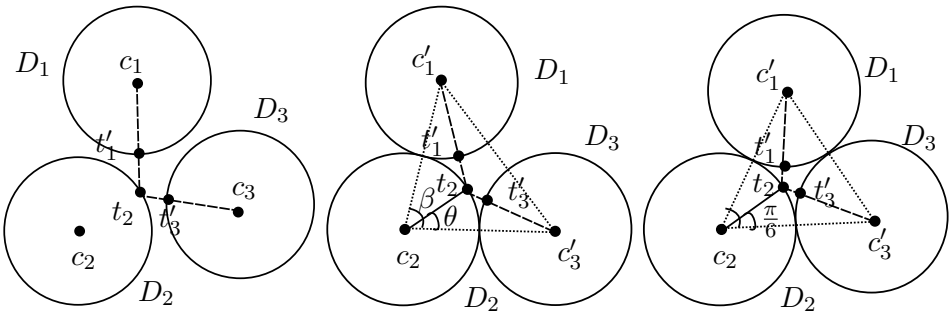

Figure 4.7: **Left:** Initial configurations of circles where $|t'_1\ t_2|, |t_2\ t'_3| < 0.4786r$. **Middle:** After rotation without changing the total distance, D_2 touches both D_1 and D_3. **Right:** The configuration where the total distance is minimum and equals to $0.4786r$.

of t_2 in terms of θ. Since $|t_1\ t_2|$ is less than $0.4786r$, we can show that the angle $\angle c'_1\ t_2\ c_2$ is greater than $\pi/2$. Using the same fact on $|t_2\ t_3|$, we can show that the angle $\angle c'_3\ t_2\ c_2$ is greater than $\pi/2$. Together with the previous inequality, we establish that t_2 is inside the triangle $\triangle c'_1\ c_2\ c'_3$. Using the same distance constraints, we can show that β is upper bounded by $\pi/2$. Moreover, since all disks are non-overlapping, it is lower bounded by $\pi/3$ (the configuration when all disks touch each other). The total distance can be expressed as:

$$r\left(\sqrt{5-4\cos(\theta)}+\sqrt{5-4\cos(\beta-\theta)}-2\right)$$

Taking the derivative of this formula with respect to θ and setting it to zero yields

that the total distance is minimized when $\theta = \beta/2$ and this value is $2r\left(\sqrt{5-4\cos(\beta/2)}-1\right)$. This value can be further minimized with respect to $\frac{\pi}{3} \leq \beta < \frac{\pi}{2}$ (i.e. $\beta = \pi/3$). Finding this value yields a configuration where all the circles touch each other and t_2 is in the middle of tangent points. In this configuration the total distance is $0.4786 \times r$ which will be used in the next Theorem to find a lower bound on the tour length. This configuration is shown in right of Figure 4.7. □

We use Lemma 3 to find a lower bound on any tour of non-overlapping, equal-sized disks in a plane. This lower bound is used for analysis of algorithms presented in subsequent sections.

Theorem 3. *Any tour τ of n disjoint, equal-sized disks of radius r, satisfies*

$$|\tau| \geq \frac{n}{2}\alpha r, \tag{4.1}$$

where $\alpha = 0.4786$ and $n \geq 3$.

Proof. Take the tour τ. It will give an order of sensors in which to visit them. Let the order be $s_1, s_2, s_3\ , \ldots, s_n$. From Lemma 3 we know that the cost of every sub-path P_i which joins s_i, s_{i+1} and s_{i+2} in τ, is lower bounded by αr for every $i \in 1, n$. Also $|\tau| \geq \frac{1}{2}\sum_{i=1}^{n}|P_i|$. Therefore $|\tau| \geq \frac{n}{2}\alpha r$. □

4.4 The General Case

In the general case, the communication disks of sensors are placed arbitrarily on the plane. Therefore the disks might be overlapping. This section presents three algorithms for this case.

Our first algorithm uses algorithms for TSPN and k-TSPN problems as subroutines. If TSPN solution is p-approximate and k-TSPN solution is q-approximate then our algorithm gives a $(p+q+5.58)$-approximate solution for the TRT problem. For example, if the inner disks of the sensors are disjoint then we can use PTAS for k-TSPN and PTAS for TSPN from [31] to get a $(7.58+\epsilon)$-approximate solution for TRT (where ϵ can be made arbitrarily small at the expense of running time). For this algorithm we assume that in the problem instance there are at least three non-intersecting outer disks.

PTAS algorithms are generally difficult to implement. Therefore, we present two algorithms which are easy to implement and have approximation ratios of $O(\frac{T_{out}}{T_{in}})$ and $O(\frac{r_{out}}{r_{in}})$ where $T_{out} > T_{in} > 0$. In practice, the ratios $\frac{T_{out}}{T_{in}}$ and $\frac{r_{out}}{r_{in}}$ are expected to be small, therefore the two algorithms are very relevant for the real world instances of TRT.

4.4.1 General Approximation Algorithm

Let C^* be the optimal tour, S_I be the set of sensors whose inner disks are visited by C^*, and S_O be the remaining sensors whose outer disks are visited by by C^*. We have $|S_I|+|S_O| = n$. Then the total cost of this tour is $OPT = |C^*|+|S_I|T_{in}+|S_O|T_{out}$ where T_{in} and T_{out} are the download times from the inner and outer disks of a single sensor. We assume that there are at least three non-intersecting outer disks in the problem instance. This is reasonable for data muling cases where the sensors are far apart. And it also gives us a lower bound on the length of C^* from Theorem 3 ($|C^*| \geq \frac{3}{2}\alpha r_{out}$).

We observe that C^* is a k-TSPN tour ($k = |S_I|$) of the inner disks and therefore, the optimal k-TSPN tour of all the inner disks is no longer than C^*. Let C_1^* be the optimal k-TSPN tour. By the previous argument, $|C_1^*| \leq |C^*|$. Suppose we can guess k ($|S_I|$), then we find an approximate k-TSPN tour rooted at base station using a q-approximation algorithm for the problem. Then the tour C_1 given by this algorithm will be of length at most $q|C_1^*|$.

Next, let C_2^* be an optimal TSPN tour of the outer disks of sensors not visited by C_1. This tour will be shorter or equal in length than the optimal tour C_{out}^* which visits all the outer disks, i.e., $|C_2^*| \leq |C_{out}^*|$. But the length of the optimal tour of the outer disks is a lower bound on the length C^* (Proposition 1). Therefore, $|C_{out}^*| \leq |C^*|$. We compute a $p-$approximate TSPN tour by using a p-approximation algorithm for the TSPN problem. If C_2 is the tour given by this algorithm, then we have $|C_2| \leq p|C^*|$

By combining the two tours we will obtain a TRT tour. If the two tours intersect then they can be combined without any additional cost. But if they do not intersect then we have to find a path to connect the two tours. Since both tours visit base station therefore the two tours can be as far as $2r_{out} - r_{in}$. We therefore know that there exist an edge with cost at most $2r_{out} - r_{in}$ which will connect the two tours.

Therefore to combine the two tours we pick the shortest edge, e, across two tours

and add it to the two tours after doubling it. This operation will increase the cost by at most $4r_{out} - 2r_{in} < 4r_{out} \leq \frac{8}{3\alpha}|OPT|$. The last part of this inequality comes from the bound on OPT from Theorem 3. We refer this tour by $C_1 \cup C_2$. Cost of $C_1 \cup C_2$ is given by

$$\begin{aligned} & |C_1| + |C_2| + 2|e| + |S_I|T_{in} + |S_O|T_{out} \\ \leq\ & q|C^*| + p|C^*| + \frac{8}{3\alpha}|C^*| + |S_I|T_{in} + |S_O|T_{out} \\ \leq\ & (p + q + \frac{8}{3\alpha})OPT \\ =\ & (p + q + 5.58)OPT \end{aligned}$$

Algorithm 2 implements the steps mentioned above. It guesses k by enumerating all possible k values and then picking up the value of k for which the total cost is minimized.

Algorithm 2 *GENERAL_APPROX_ALGORITHM*

```
TOUR ← φ
minCost ← ∞
for k = 1 to n do
    C1 ← k-TSPN tour of the inner disks
    S ← { sensors visited by C1}
    C2 ← TSPN tour of the outer disks of the sensors not included in S
    e ← shortest edge joining C1 and C2
    cost ← |C1| + |C2| + 2|e| + kTin + (n − k)Tout
    if minCost > cost then
        TOUR ← C1 ∪ C2
return TOUR
```

Mitchell presents PTAS algorithms for both TSPN and k-TSPN problems for disjoint neighborhoods [32]. If we use these algorithms for the case when the outer disks are disjoint, then we get $p = (1+\epsilon/2)$ and $q = (1+\epsilon/2)$. In that case our algorithm yields a $(7.58+\epsilon)$-approximation factor. If the inner disks are disjoint, we can use the constant factor approximation algorithm for the outer disks [31] to find a TSPN tour ($p = 11.15$) and the PTAS for inner disks to find a k-TSPN tour which yield to an approximation algorithm with factor $(16.73 + \epsilon)$.

4.4.2 $O(\frac{T_{out}}{T_{in}})$-approximation

Algorithm presented in this section is appropriate for the case when the download times of the inner and the outer disks are comparable. For this scenario, we show that the strategy of visiting just the outer disks of the sensors yields an $O(\frac{T_{out}}{T_{in}})$-approximation for TRT.

First, we use a TSPN algorithm (e.g. [31]) to find a TSPN tour C_{out} of all the outer disks. Let p be the approximation factor for this algorithm and $p \geq 1$. Since the tour visits the outer disks, the robot downloads data from the sensors with the download speed of T_{out}. Therefore, the approximation factor of the algorithm is -

$$\begin{aligned} \frac{|C_{out}| + nT_{out}}{OPT} &\leq \frac{|C_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \\ &\leq \frac{p|C^*_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \\ &\leq p\frac{|C^*_{out}| + nT_{out}}{|C^*_{out}| + nT_{in}} \leq p\frac{T_{out}}{T_{in}} \end{aligned} \tag{4.2}$$

The first inequality in Equation 4.2 comes from Proposition 2. TSPN approximation directly yields the second inequality. Therefore, our algorithm has an approximation factor of $O(\frac{T_{out}}{T_{in}})$.

4.4.3 $O(\frac{r_{out}}{r_{in}})$-approximation

For some problem instances the ratio of radii may be better than the ratio of download times. In such cases if the ratio of radii is small, we can find an efficient algorithm with the approximation factor of order $O(\frac{r_{out}}{r_{in}})$.

In this algorithm, first we compute a maximal non-overlapping set I of the outer disks. For this we use Algorithm 3. Then we find a TSPN tour C^I_{out} of the disks in I. For each disk $A \in I$, we define the set of sensors whose outer disks intersect with A as S_A. Next, we will show that how we can extend C^I_{out} such that it visits all the inner disks of the sensors in S_A. Also, we assume that $|I| \geq 3$.

Let D be a disk of radius $2r_{out}$ and is co-centered with A. All the sensors in S_A lie on or in D. We traverse D in concentric circles which are distance r_{in} apart as shown in Figure 4.8. This will ensure that all the inner disks of the sensors in S_A are visited.

Algorithm 3 *PARTITION_ALGORITHM*(S)

$I \leftarrow \phi$

while $S \neq \phi$ **do**

Pick a outer disk D of a sensor $\in S$.

$I \leftarrow I \cup D$

$S_D \leftarrow$\{ sensor with outer disk $D' \in S : D' \cap D \neq \phi$\}

$S \leftarrow S - S_D$

return MIS set I and all partitions S_D.

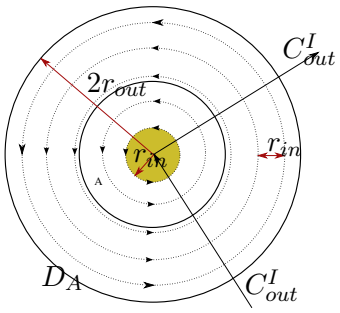

Figure 4.8: In all inner-disk visits, the algorithm chooses to sweep the $2r_{out}$ size disk centered at A in concentric circles which are r_{in} apart.

Let d_A be the extra distance traveled in this process and let $k = \lfloor \frac{2r_{out}}{r_{in}} \rfloor$. Then, $d_A = \sum_{i=1}^{k} 2\pi i r_{in} = \pi r_{in} k(k+1) = 2\pi r_{out}(k+1)$. The cost of this tour is $|C^I_{out}| + md_A$ $= |C^I_{out}| + 2m\pi r_{out}(k+1)$, where $m = |I|$. This gives us the approximation ratio

$$\begin{aligned} \frac{|C^I_{out}| + md_A + nT_{in}}{|{C_{out}}^*| + nT_{in}} &\leq \frac{|C^I_{out}| + md_A}{|{C_{out}}^*|} \\ &= \frac{|C^I_{out}|}{|{C_{out}}^*|} + \frac{2m\pi r_{out}(k+1)}{|C^*_{out}|} \\ &\leq p + \frac{2m\pi r_{out}(k+1)}{\frac{m}{2} r_{out}\alpha} \\ &= p + \frac{4\pi(k+1)}{\alpha} \end{aligned} \tag{4.3}$$

In the second inequality we use p as the approximation ratio for the TSPN algorithm and the lower bound on C^*_{out} is obtained from Theorem 3. Finally, this gives us an $O(\frac{r_{out}}{r_{in}})$ approximation algorithm under the requirement that $|I| \geq 3$.

4.5 Simulations

In this section we compare the tour of outer disks, the tour of inner disks and the tour of centers of the sensors using simulations. The setup consists of 100 sensors deployed uniformly at random on a 100×100 grid. The radii of the outer disks was set to 10 units for all the trials. Similarly, the download time T_{out} for the outer disks was fixed to 10 units.

First we varied the inner disk radii r_{in} from 1 to r_{out} and computed the inner disks tour cost. The inner disk download time T_{in} was fixed to 5 units. We performed 100 trials for each r_{in} value and reported the average tour cost. Top:Figure 4.9 shows the average tour cost for different values of the r_{in}. The vertical bars in the figure shows the standard deviation of the tour cost. It was surprising to observe that the inner disk tour's cost increased with increasing r_{in}. On further analysis, we observed that as r_{in} increases the size of the independent set I used to compute TSPN tour decreases (see Bottom:Figure 4.9). When the intersections are few the approximation algorithm yields simple tours and the computed tours are closer to optimal tour. But when the intersections are significant, although the tour cost is bounded by a constant factor, the tours computed are somewhat complex and are relatively less closer to optimal.

In the next experiment we change T_{in} while keeping r_{in} fixed to 5. The inner disk tour cost varies linearly with T_{in} (Figure 4.10). The dashed line in the figure represents the length of the outer disk tour. Although the length of optimal outer disk tour is a lower bound on cost of any inner disk tour (Proposition 1), but the dashed line shown in the figure is the approximate solution to the outer disk tour.

In the final experiment we fixed both r_{in} and T_{in} to 5. We increased the number of sensors on the grid and computed the tours. Figure 4.11 shows the plot of the experiment. We observe that after increasing number of sensors significantly, the outer disk tour length does not change much. The TSPN algorithm used to compute the tour [31] computes the independent set I. After a while the size of I does not change with increase in the number of sensors and hence the tour length also does not vary much.

In the following chapter we consider the energy issues of the robot and present energy harvesting ideas to increased the operational life of the robot.

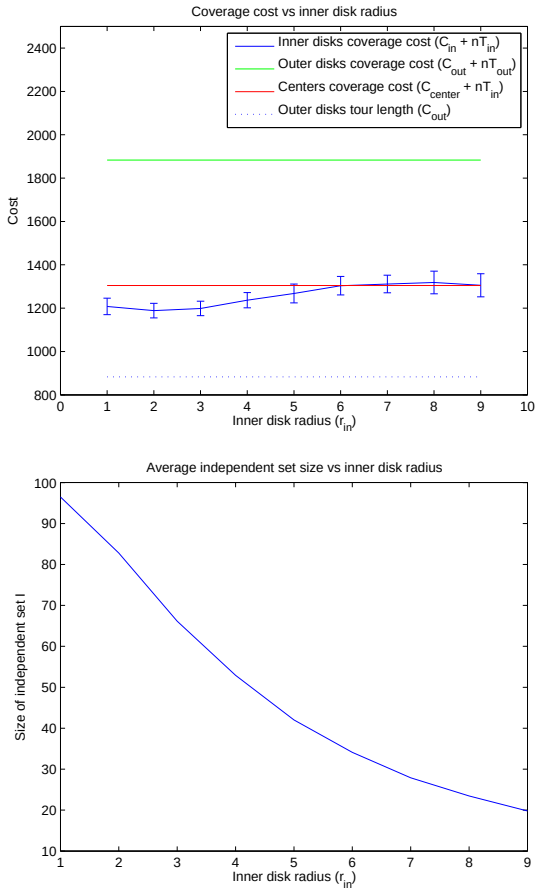

Figure 4.9: **TOP**: This figure shows the change in tour cost with the change in inner disk radius r_{in}. Interestingly the inner disk tour cost increased with increasing r_{in}. **BOTTOM**: This figure shows that the size of independent set I decreases as r_{in} increases. The independent set I is used to compute the TSPN tour using algorithm in [31]. This attributes to the increase in the inner disk tour cost in the TOP figure.

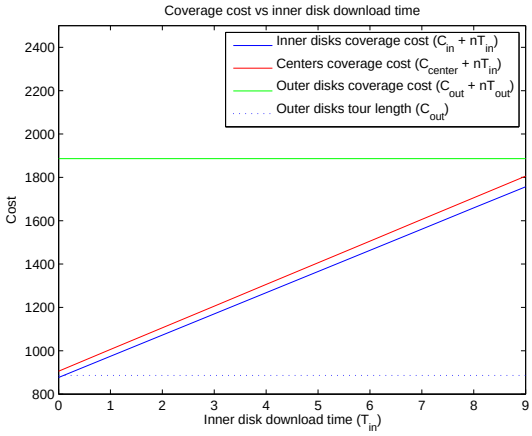

Figure 4.10: The increase in inner disk and center tour cost is linear with increase in T_{in} as expected.

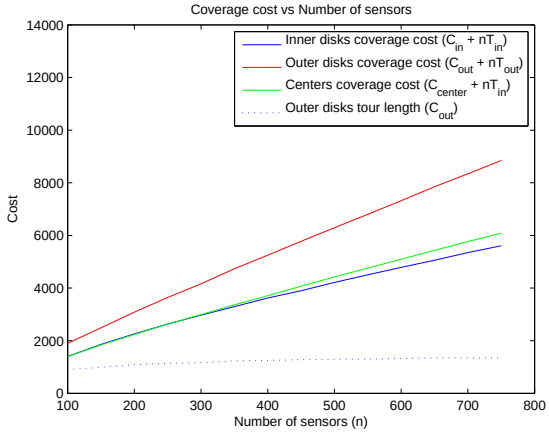

Figure 4.11: As number of sensors are increased tours costs also increase. This is expected as the download time increases linearly with the number of sensors.

Chapter 5

Energy Harvesting

In the data muling experiment in Chapter 3 we observed that the robot's batteries drain out very quickly. Current custom made robots available in the market can run for around 2 hours at most on battery power. This time further shortens in rough terrains with long grass, sand or slopes. This limitation of energy at the robot puts a limit on the duration of a data muling operation. Therefore it is necessary to improve the lifetime of the robot.

One solution is to put more batteries. But every robot has a payload limit and therefore only a limited number of batteries can be carried by a robot. Moreover, increase in payload on the robot also increases the consumption of energy. Therefore it is not viable to keep adding batteries to the robot.

Other option is to either recharge/replace the batteries or replenish the robot's energy by harvesting energy from the nature. Since we want to reduce the human intervention as much as possible we started looking into the option of harvesting energy. Energy harvesting for data muling systems is a viable option because most of data muling systems are deployed outdoors where ample natural resources like sun, wind are available. For our system we started looking into solar energy harvesting with a solar panel on our outdoor robot.

In this chapter we first present an experimental model of energy consumption and harvesting as a function of environmental parameters. Next, we study the problem of finding an energy efficient trajectory for robot to go from a source point to a destination point such that it collects maximum energy within a given time budget. For this problem

we present an optimal algorithm based on dynamic programming which uses the model developed here.

5.1 Initial Experiments

We started with an experiment with solar panel to find out its power generation capacity in various conditions. We used a voltage and current measuring circuit and connected it to solar panel power output. From the circuit we can obtain current and voltage across the solar panel.

First we kept the solar panel in an area with no direct sunlight. The power measured by the circuit was nearly 0W. After that we kept the solar panel in sunlight. For this case we measured power for two different configurations of the solar panel: a) oriented towards the sun b) lying parallel to the ground. Figure 5.1 shows the plot of power vs time for the solar panel kept in sunlight. When the solar panel was oriented towards the sun we obtained 19.3W of mean power. However when the solar panel was lying parallel to ground we obtained a mean power of 1.3W which is negligible. The power drop is significant considering that in both cases the solar panel was in sun. These results suggests that the power generated by the solar panel can be modeled as a binary function of the conditions: If solar panel is oriented towards sun in a sunlit area then we obtain 19.3W power else we obtain no/negligible power.

In our second experiment we looked into power consumption characteristics of our outdoor robot Husky (Figure 5.2). Husky is manufactured by ClearPath Robotics. It is a 6×6 differential drive system. It can carry a payload of 40kgs and have a maximum speed of 1.5m/s. Husky has two 12V, 44AH batteries which provide 2 hours of operation. Husky provides a sturdy frame and we mounted a solar panel on top of it as shown in the figure.

We tested Husky's power consumption at various speeds. Figure 5.3 shows the results. Without solar panel as payload we observed that the power consumption is a linear function with Husky's speed. When we added the solar panel (without energy harvesting) the power consumption was still linear but shifted vertically by around 15W on average. The experiment verifies our claim that we can not keep adding batteries to a robot because it not only limits the payload capacity of robot but also increases the

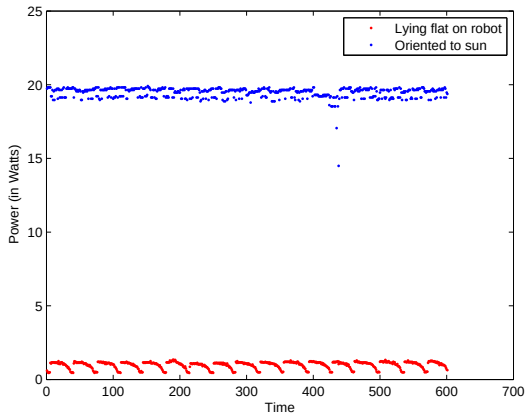

Figure 5.1: Power produced by solar panel

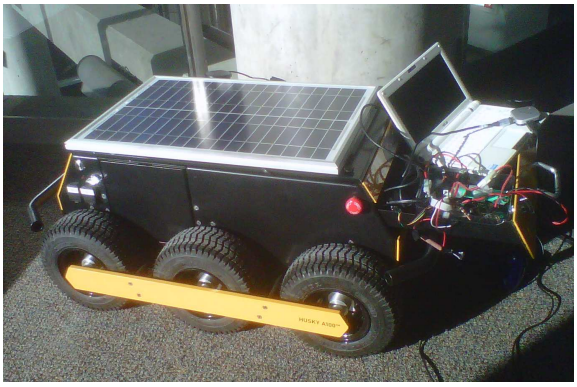

Figure 5.2: Husky robotic platform with solar panel mounted on it

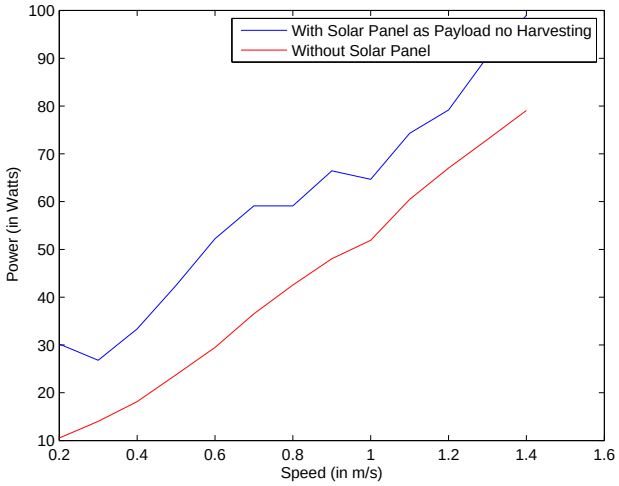

Figure 5.3: Power consumption by Husky at various speeds

power consumption at the robot. However, in case of solar panel since solar panel can harvest around 19W it does not add any extra constraint on robot's power requirement.

These experiments also gives two important parameters necessary for modeling our system. First is e_h, the amount of energy harvested by solar panel per unit time. For our case $e_h = 19.3$. Next is e_s which is the amount of energy consumed in travelling at constant speed per unit time. If we fix robot's speed to 0.4m/s then from the power consumption plot we get $e_s = 30$.

In next section we describe the system model and define a new problem where the robot's goal is to go from a source point to a destination point such that it collects maximum energy within a given time budget.

5.2 Problem Definition

To model the environment we first place a grid on the environment. We assume that we are given a solar map of the environment. A solar map has a binary value information of sun (sunny or dark) for each cell in the grid (for eg. see Figure 5.4). A value 1 tells that the cell is sunny and a value 0 tells cell is either completely shaded or partially shaded. It is reasonable to assume value 0 for partially shaded cells because solar panel throughput

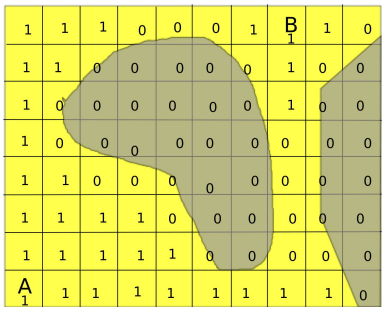

Figure 5.4: A solar map model. Yellow color is sunny area and gray is shaded.

decreases significantly if it is partially covered. Moreover by modeling partially shaded cells as 0 we consider the worst case situation.

A solar map can be manually constructed based on the elevation of the sun and position of the static objects like buildings or trees. Or we can learn this map using reinforcement learning techniques. A solar map changes with time because of the change in position of the sun, but for a short duration of time it does not change significantly. With this model we pose the energy harvesting problem as follows.

Energy Harvesting Problem: Given a source (say A) and a destination point (say B) and a solar map find a trajectory for robot to go from A to B such that the energy remaining on the robot is maximized and the total time taken to reach B is within given time budget T.

Energy change on the robot is due to two factors: a) the energy spent by the robot in travel and communication and b) the energy harvested by the solar panel on the robot. Let the total energy spent by robot in travel and communication from time 1 to t be E_s^t and the total energy harvested by robot from time 1 to t be E_h^t. Then for time budget T we want to maximize $E_h^T - E_s^T$ such that robot travels from A to B within T time.

As mentioned before we model the environment as a grid. The solar map contains information whether sun is available or not in a cell. A robot can collect the solar energy while travelling or by stopping at a cell for some time. We assume that the robot stops at the center of the grid cell to harvest solar energy.

In the following section we will present an algorithm based on dynamic programming to find out the energy optimal trajectory for robot for a given map, source and

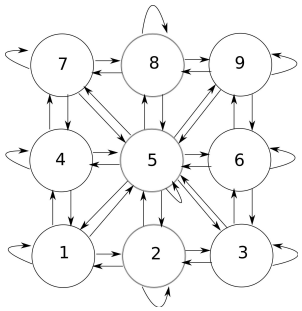

Figure 5.5: State transition diagram for a 3×3 grid

destination cells, and a time budget.

5.3 Algorithm

We represent each cell of the grid as a state. Since a robot can go from one cell to any of the adjoining cell or can stay in the same cell, we connect the corresponding states by an edge in state transition diagram (see Figure 5.5). We define the distance between the two cells as the edge weight on the edge connecting respective states. We represent the set of states by P. For a state $p \in P$ we represent the set containing all neighboring states of p by N_p.

As mentioned in previous section e_s is the energy spent per unit time by the robot in travelling and e_h is the energy harvested per unit time by the robot in a sunny area. Let P_t be the state of robot at time t. We overload P_t and also use it to represent the total energy of the robot at time t. Let $map(P_t)$ be the function which returns 1 if state P_t is sunny and returns 0 otherwise. Similarly let $dis(P_t, P_u)$ be the function which returns distance between the states P_t and P_u. Then,

$$P_{t+1} = P_t + e_h * map(P_t) - e_s * dis(P_{t+1}, P_t) \tag{5.1}$$

Equation 5.1 computes the total energy remaining on the robot after robot moves from state P_t to state P_{t+1}. However we still do not know how to compute the next state. We only know our initial state and our final state.

We observe that the problem follows a suboptimal property. Suppose $P_1, P_2, \ldots,$ P_{T-1}, P_T is the optimal path to reach B from A in T time units. Then $P_1, P_2, \ldots, P_{T-1}$

should be the optimal path to reach P_{T-1} in $T-1$ time units otherwise we can replace it with the optimal path of $T-1$ time units and get a better path than $P_1, P_2, \ldots, P_{T-1}, P_T$. Using this fact we constructed a dynamic programming table with entry $E[a,t]$ representing the energy of robot at state a and time t.

$$E[a,t] = \begin{cases} -\infty & \text{if } t = 1 \text{ and } a \neq A \\ 0 & \text{if } t = 1 \text{ and } a = A \\ \max_{b \in N_a}(E[b,t-1] + e_h * map(b) - e_s * dis(a,b)) & \text{otherwise} \end{cases} \quad (5.2)$$

$$P_t = \begin{cases} B & \text{if } t = T \\ \arg\max_{b \in N_{P_{t+1}} and E[b,t] \geq E_{thresh}}(E[b,t] + e_h * map(b) - e_s * dis(P_{t+1}, b)) & \text{otherwise} \end{cases} \quad (5.3)$$

Note that the states in which $E[b,t] < E_{thresh}$ are not considered in computing the path. This is due to the fact that the robot should always have at least E_{thresh} energy to continue being in operation.

We ran the algorithm with various system parameters. In first case we gave enough time budget so that the robot can reach point B. In this case robot tried to keep on sunny area as much as possible (See Top Left of Figure 5.6). When the time budget was increased the computed trajectory completely avoided the dark areas (Top Right of Figure 5.6). However, even with higher time budgets, if the energy spent in travelling is increased the robot goes straight from point A to point B while stopping in sunny areas to harvest energy (Bottom of Figure 5.6). Numbers in Figure 5.6 represent the position of robot at that time unit.

This chapter provides a starting point for solar harvesting while performing navigation tasks. The algorithm presented can be used as a subroutine for the DGP problem and utilized after the order of the nodes is chosen. We leave the more challenging problem of finding an energy efficient tour of n points which maximizes the overall energy remaining on the robot for a given time budget for future work.

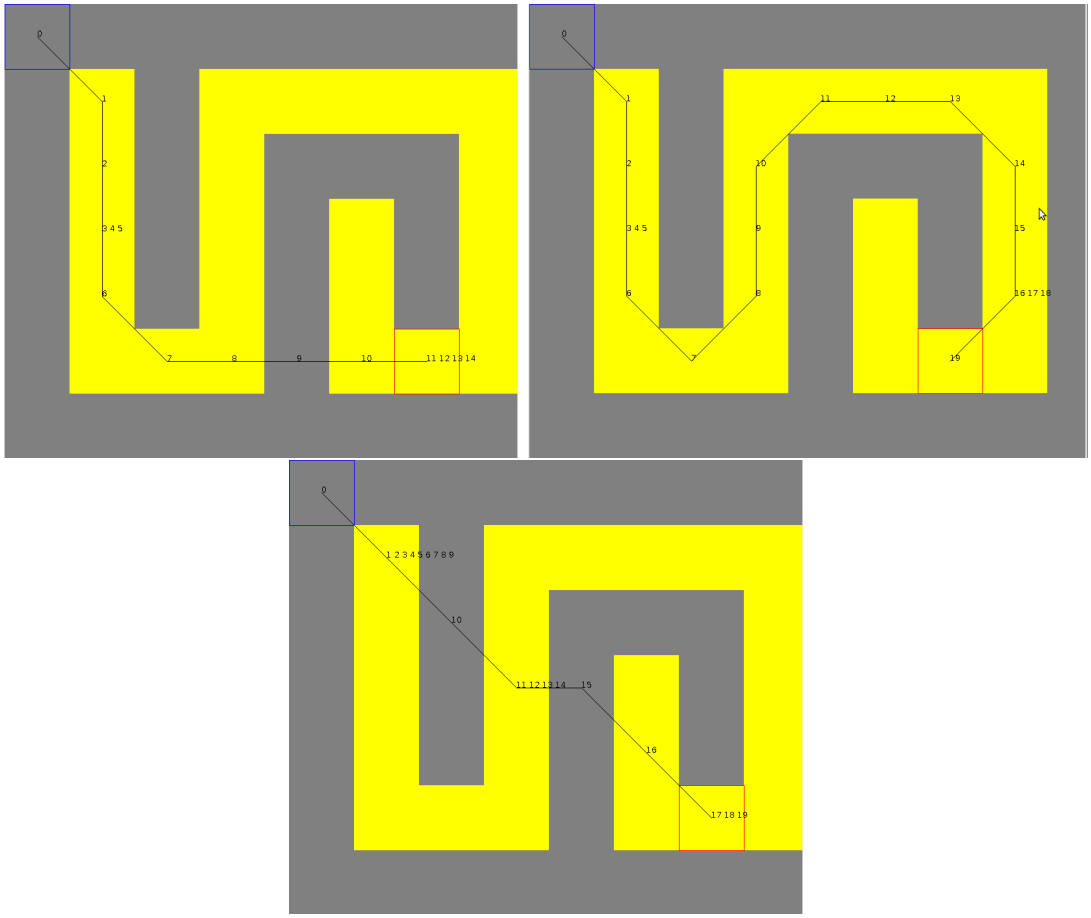

Figure 5.6: Simulation results **Top Left:** Sufficient time budget **Top Right:** Extra time budget **Bottom:** Extra time budget with higher travelling cost

Chapter 6

Conclusion

In this work, we first studied a path planning problem, the Data Gathering Problem (DGP), which arises in scenarios where robots act as *data mules* to download data from stationary wireless devices. The problem differs from the well-known TSP problem due to the fact that downloading data takes time. Therefore, the cost of a tour is affected by not only the travel time but also the download time as well as the number of downloads.

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. This case is applicable in scenarios where the robots are restricted to move along a predetermined path such as a railroad track. 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.

We implemented the algorithm on a data muling system. Results from a field experiment demonstrated the effectiveness and utility of the system. Results also suggested the need of a better communication model. Further experiments suggested that sensor nodes communication is better represented by a Two-Ring communication model: two concentric disks around each sensor with different download times.

For Two-Ring communication model we introduced a novel path planning problem where the mobile entity must decide which disk to visit for each sensor, and spend the corresponding time to download its data. We called this problem as Two-Ring Tour

(TRT) problem. For the general case of TRT, we first presented an algorithm whose approximation is a function of the approximations to k-TSPN and TSPN problems. We also presented two polynomial time algorithms whose performance ratios are proportional to the ratio of the two radii or the ratio of the download times. As demonstrated in motivating examples, these two parameters are usually small in some practical scenarios.

Finally we initiated the study of harvesting energy from Nature to improve life time of the data muling system. In Chapter 5, we presented preliminary work on solar energy harvesting. We modeled the environment based on data observed from experiments on solar panel and Husky robot. We also studied the energy harvesting problem: Find an energy efficient trajectory for a robot to go from a source point to a destination point such that it collects maximum energy within a given time budget. For this problem we presented an optimal algorithm based on dynamic programming.

There are many interesting areas for future work. For DGP, the algorithms presented in the thesis can accommodate obstacles as long as they do not intersect with the communication disks. When the communication disks of sensors are not obstacle free, a new algorithm for DGP will be required. Our current design assigns each sensor to a single robot. It may be beneficial to allow the robots to communicate and collaborate with each other during the data muling task. Our future work will include exploring this multi-robot communication/cooperation aspect of data muling. In case of the TRT problem, we will explore more realistic communication models such as multiple rings or a continuous function of distance. Energy harvesting ideas presented in this work are preliminary. We would like to extend these ideas to provide more refined models for energy harvesting. It will assist in solving the bigger problem of sustainable data muling. Also we will study the energy harvesting problem where the goal is to compute a tour of n points which maximizes the overall energy remaining on the robot within a given time budget.

References

[1] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based sensor node energy estimation. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 409–410. ACM, 2007.

[2] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *IEEE INFOCOM*, pages 1380–1387, 2001.

[3] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325 – 349, 2005.

[4] Vineyard uses sensor network to fine-tune irrigation. NetworkWorld, 2009.

[5] 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.

[6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *ACM SIGOPS operating systems review*, 36(5):96–107, 2002.

[7] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 68–77, Atlanta, GA, USA, 2002. ACM.

[8] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor networks. In *Proceedings of the 2nd International Conference on Information Processing in Sensor Networks*, pages 129–145, Palo Alto, CA, USA, 2003. Springer-Verlag.

[9] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. pages 244–257, Marina del Rey, CA, USA, 2005. Springer.

[10] L. Boloni and D. Turgut. Should I send now or send later? A decision-theoretic approach to transmission scheduling in sensor networks with mobile sinks. *Wireless Communications and Mobile Computing*, 8(3):385–403, 2008.

[11] G. Anastasi, M. Conti, E. Monaldi, and A. Passarella. An adaptive data-transfer protocol for sensor networks with Data Mules. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–8, Helsinki, Finland, 2007.

[12] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. pages 1096 –1102, Marrakech,Morocco, 2008.

[13] Jian Ma, Canfeng Chen, and Jyri P. Salomaa. mWSN for large scale mobile sensing. *J. Signal Processing Systems*, 51(2):195–206, 2008.

[14] E. Ekici, Y. Gu, and D. Bozdag. Mobility-based communication in wireless sensor networks. *IEEE Communications Magazine*, 44(7):56, 2006.

[15] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.G. Lee. Partitioning based mobile element scheduling in wireless sensor networks. In *Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 386–395, Santa Clara, CA, USA, 2005.

[16] J. Xing, H. Wang, K. Han, D.A. Ray, C.H. Huang, L.G. Chemnick, C.B. Stewart, T.R. Disotell, O.A. Ryder, and M.A. Batzer. A mobile element based phylogeny of Old World monkeys. *Molecular Phylogenetics and Rvolution*, 37(3):872–880, 2005.

[17] Aman Kansal, Arun A. Somasundara, David D. Jea, Mani B. Srivastava, and Deborah Estrin. Intelligent fluid infrastructure for embedded networks. In *International Conference on Mobile Systems, Applications, and Services*, pages 111–124, Boston, MA, USA, 2004.

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

[19] A. Gasparri, B. Krishnamachari, and G.S. Sukhatme. A framework for multi-robot node coverage in sensor networks. *Annals of Mathematics and Artificial Intelligence*, 52(2):281–305, 2008.

[20] K. Williams and J. Burdick. Multi-robot boundary coverage with plan revision. In *IEEE International Conference on Robotics and Automation*, pages 1716–1723, 2006.

[21] Y. Tirta, Zhiyuan Li, Yung-Hsiang Lu, and S. Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. pages 515–519, Chicago, IL, USA, 2004.

[22] F.J. Wu, C.F. Huang, and Y.C. Tseng. Data gathering by mobile mules in a spatially separated wireless sensor network. In *International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 293–298, Taipei,Taiwan, 2009. IEEE Computer Society.

[23] 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.

[24] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. pages 2091–2098, Orlando, FL, USA, 2006.

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

[26] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[27] N. Christofides. Worst case analysis of a new heuristic for the traveling salesman problem. Technical report, Graduate School of Industrial Administration, 1976.

[28] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[29] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

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

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

[32] J. S. B. Mitchell. A ptas for tsp with neighborhoods among fat regions in the plane. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–18, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[33] K. Elbassioni, A. V. Fishkin, and R. Sitters. On approximating the tsp with intersecting neighborhoods. In *Proc. 17th Annu. Internat. Sympos. Algorithms Comput*, 2006.

[34] M. Rahimi, H. Shah, G.S. Sukhatme, J. Heidemann, and D. Estrin. Studying the feasibility of energy harvesting in a mobile sensor network. In *IEEE International Conference on robotics and Automation*, volume 1, pages 19–24. Citeseer, 2003.

[35] P. Tompkins, D. Wettergreen, T. Stentz, and W. Whittaker. SUN-SYNCHRONOUS NAVIGATION: TERRESTRIAL DEMONSTRATION AND ITS FUTURE FOR PLANETARY EXPLORATION. 2008.

[36] Pratap Tokekar, Nikhil Karnad, and Volkan Isler. Energy-optimal velocity profiles for car-like robots. In *ICRA*, 2011(submitted).

[37] D. Bhadauria and V. Isler. Data gathering tours for mobile robots. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3868–3873, St. Louis, MO, USA, 2009. IEEE Press.

[38] Greg N. Frederickson, Matthew S. Hecht, and Chul E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978.

[39] Deepak Bhadauria, Onur Tekdas, and Volkan Isler. Robotic Data Mules for Collecting Data over Sparse Sensor Fields. *Journal of Field Robotics*, 2010.