DATA PRESERVATION IN ROBOTIC SENSOR NETWORKS: A BUDGET-CONSTRAINED COVERING SALESMAN APPROACH

_____

A Thesis

Presented

to the Faculty of

California State University, Dominguez Hills

_____

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

_____

by

Soham Patil

Spring 2024

THESIS: DATA PRESERVATION IN ROBOTIC SENSOR NETWORKS: A BUDGET CONSTRAINED COVERING SALESMAN APPROACH

AUTHOR: SOHAM PATIL

APPROVED:

_____
Bin Tang, Ph.D.
Thesis Committee Chair

_____
Ali Jalooli, Ph.D.
Committee Member

_____
Sanaz Rahimi Moosavi, Ph.D.
Committee Member

ACKNOWLEDGEMENTS

With profound gratitude, I acknowledge the invaluable guidance and mentorship of Dr. Bin Tang. His infectious enthusiasm for research and visionary spirit fostered a fun and engaging learning environment, propelling me forward throughout this journey. Dr. Tang's reminder that "research is a journey" provided much-needed encouragement, helping me stay focused and motivated to explore new avenues throughout this process. This research thesis would not have come to fruition without his unwavering support and constant encouragement.

I extend my sincere thanks to Dr. Sanaz Moosavi and Dr. Ali Jalooli for the insightful corrections and guidance on report and thesis writing which proved invaluable throughout this undertaking.

Finally, I express my deepest appreciation to my family, especially my parents, my uncle and aunt and my girlfriend. Their unwavering support, encouragement, and faith in me provided the bedrock upon which I built my perseverance and success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This report centers on robotic sensor networks (RSNs), where mobile data collectors, or robots, are deployed in sensor fields to gather data from sensor nodes and return to a base/charging station. We investigate a novel algorithmic problem termed DCR (data collection in RSNs). The objective of DCR is to dispatch a robot into the sensor field to gather the maximum number of data packets before exhausting its battery and returning for recharging. Despite extensive research on enhancing data collection in RSNs, little attention has been given to the limited battery power of data-collecting robots. We demonstrate that the DCR framework revolves around new graph-theoretical problems, termed the Budget-Constrained Traveling Salesman Problem (BC-TSP) and Budget Constrained Covering Salesman Problem (BC-CSP). To address the BC-TSP, we develop a set of algorithms including an Integer Linear Programming (ILP)–based optimal solution, three iterative greedy algorithms, and a multi-agent reinforcement learning (MARL) algorithm. Further we apply these algorithms to solve Budget Constraint Covering Salesman Problem Approach to show that it can collect more data packets than BC-TSP approach but depletes the network longevity. Through comprehensive simulations employing real measurements of battery power and robot mobility, for the BC-TSP our results indicate that: a) our algorithms outperform existing approaches by collecting 25% more data packets with the same battery power, and b) MARL performs competitively compared to handcrafted greedy algorithms. In the BC-CSP approach, our greedy algorithms demonstrate the capacity to gather 40% more data packets compared to the BC-TSP approach greedy algorithms.

Keywords – Data collection, robotic sensor networks, integer linear programming, prize-collecting covering salesman problem, multi-agent reinforcement learning.

CHAPTER 1

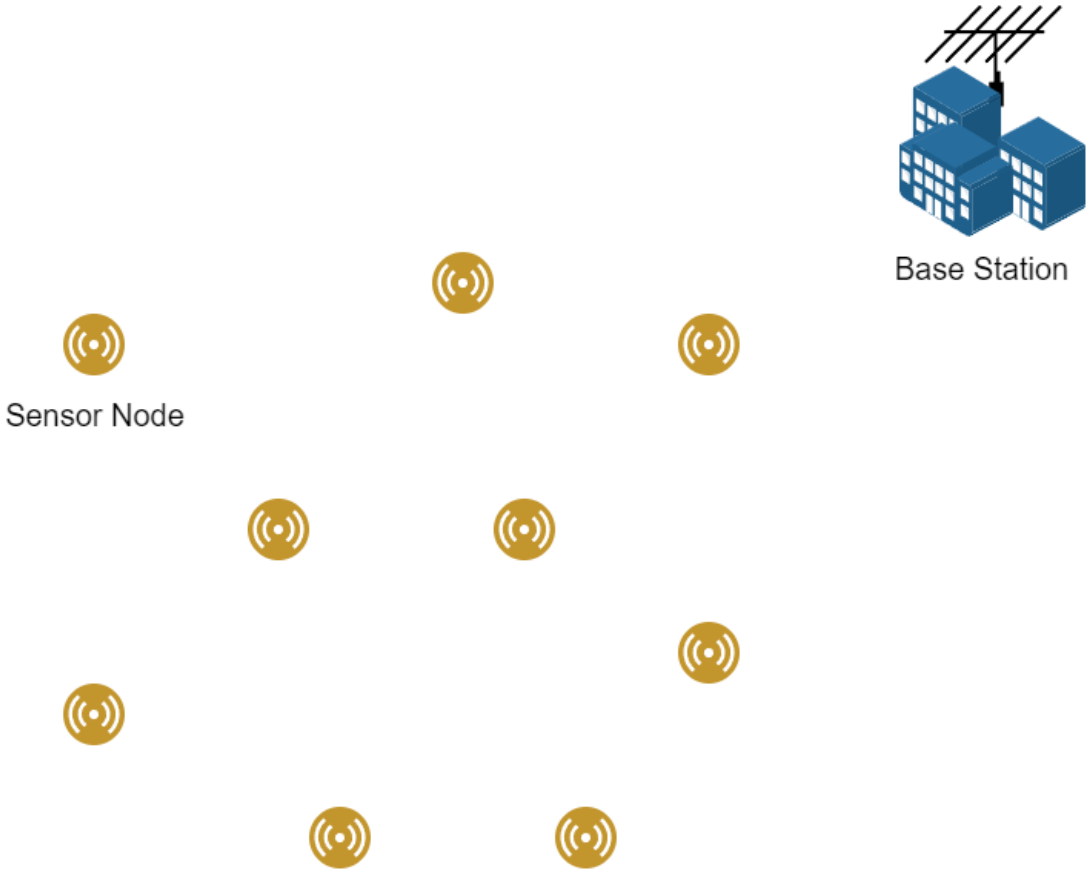
INTRODUCTION

Background and Motivation. Since its inception in the early 1990s, wireless sensor network (WSN) research has transitioned from laboratory settings to practical applications across various domains such as military, health, environmental monitoring, industrial processes, and urban infrastructure, facilitating diverse data collection from the physical world (Rawat et al., 2014; Rahmati & Pompili, 2019). Recent advancements in artificial intelligence (AI) and machine learning (ML), particularly in deep reinforcement learning, have propelled significant progress in robotic research, leading to the emergence of various robotic applications (Garaffa et al., 2023; Kober et al., 2013). Notably, considerable efforts have been directed towards leveraging mobile robots to enhance the performance and efficiency of wireless sensor networks, particularly in environment monitoring, intrusion detection, and search and rescue operations. This integration of mobile robots with wireless sensor networks is termed robotic sensor networks (RSNs). The two main tasks that mobile robots execute to improve the performance and functions of RSNs are data collection and network maintenance (e.g., wirelessly recharging sensor nodes). Figure 1 shows the basic sensor network entities. By dispatching the battery-rechargeable robots into the sensor field to collect the data and bring it back to the base station, the energy bottleneck of the sensor networks is migrated from the sensor nodes to the mobile robots. Consequently, the lifetime of RSN is greatly improved compared to the traditional sensor networks, wherein sensory data is transmitted back to the base station in an energy-consuming multi-hop manner. In multi-hop wireless communications, as sensor nodes close to the base station are responsible for forwarding data to the base station from nodes that are farther away which is depicted in Figure 2, they could quickly deplete their battery power. Thus, the closer a sensor node is to a

sink, the faster its battery runs out. Besides, the precious battery powers should be utilized for sensing, computing, and actuation activities, which are the purposes of sensor network design and deployment, instead of routing.

Figure 1

Sensor Network Diagram



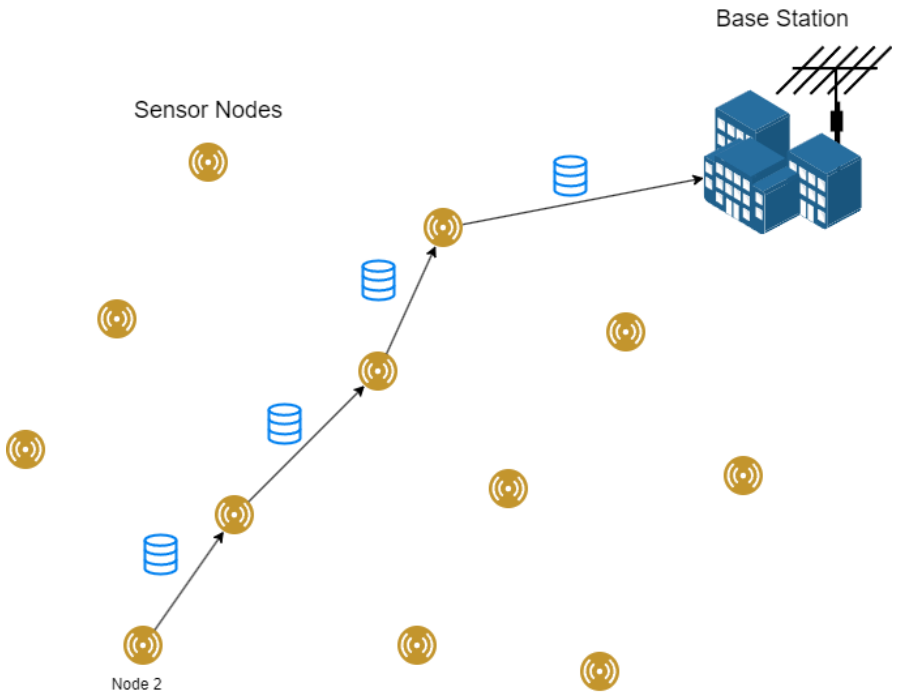
Additionally, many challenging and inaccessible environments, such as those found in seismic activity monitoring (McNulty et al., 2022) or deep-water exploration (Coutinho et al., 2019; Phillips et al., 2017), necessitate the deployment of autonomous underwater vehicles (AUVs) or robots for data collection. Hence, data collection remains a crucial task for mobile robots in modern RSNs. Considerable research efforts have been dedicated to addressing various objectives related to data collection within RSNs, which can be broadly classified into two

categories. The first category aims to optimize resource provisioning objectives of data collection. This includes minimizing the length of data-collecting tours for robots (Ma et al., 2013; Yang & Wang, 2016), reducing energy consumption of wheeled mobile robots (Liu & Sun, 2014), or minimizing latency in drones during search-and-reconnaissance operations (Kim et al., 2017; Xue et al., 2014). Additionally, efforts have focused on maximizing network lifetime and implementing path planning strategies to achieve full connectivity for disconnected sensor networks. The second category aims to achieve admission control objectives, where not all sensor nodes can be visited to collect all sensory data. This involves maximizing network utility, measured in terms of collected data packets, through joint wireless energy replenishment and mobile data gathering (Guo et al., 2014; Wang et al., 2016). Other objectives in this category include collecting as much data as possible within a specified time duration or retrieving all sensed data within a given deadline using rendezvous points (Salarian et al., 2014; Wang & Chen, 2019).

However, most of the existing research assumes that robots possess sufficient battery power, if needed, to traverse the entire RSN field and collect all sensory packets before returning to the depot for recharging. However, given that robots have limited yet rechargeable battery power, there exists a possibility that they can only travel a finite distance before battery depletion. For instance, Xiao et al. illustrate that the propulsion energy, responsible for driving robots forward, for various robotic rovers typically ranges from tens to hundreds of watt-hours (Wh), limiting their traversal distance to tens of kilometers. In large-scale RSNs, when robots cannot collect all sensory data before returning to the charging station, it becomes crucial to efficiently schedule battery-constrained robots to gather as much valuable sensory information as possible. This optimization is necessary to enhance the performance of RSN applications.

Figure 2

Multi-Hop Data Transmission

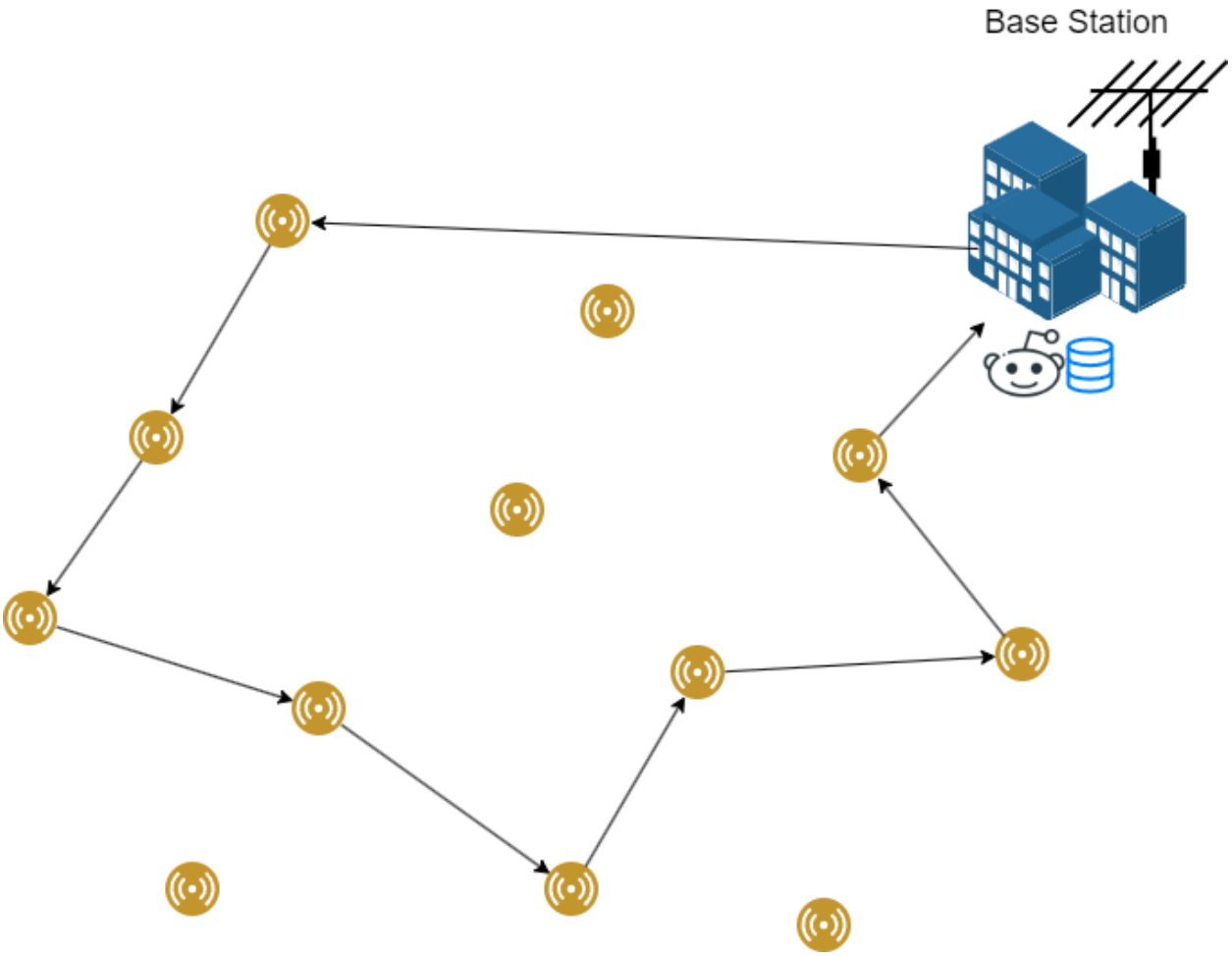


To address this challenge, we introduce a novel algorithmic framework named DCR: Data Collection in RSNs. The objective of DCR is to dispatch a robot, equipped with limited battery power, into the sensor field of an RSN composed of sensor nodes, each generating varying numbers of data packets. Figure 3 shows the Robotic Sensor Network. The aim is to maximize the collection of data packets by the robot before its battery power is depleted, necessitating its return to the depot for recharging. In the process of solving DCR, we identify a new graph-theoretical problem termed the Budget-Constrained Covering Salesman Problem (BC-CSP), which represents a variation of the Covering Salesman Problem. In this problem the only difference is that the robot will be having a transmission range. Once it goes to a specific sensor node it can collect data from all other nodes that fall within the transmission range as shown in Figure 4. To address the BC-CSP, we develop a suite of algorithms, an iterative greedy

algorithm, and a multi-agent reinforcement learning (MARL) algorithm. Through extensive simulations utilizing real-world measurements of battery power and robot mobility, our findings demonstrate that: a) our algorithms surpass existing approaches by collecting 25% more data packets with equivalent robot battery power, b) MARL performs way better compared to manually crafted greedy algorithms. In contrast, existing approaches to data collection, such as the covering salesman approach, risk depleting the battery power of certain sensor nodes within approximately 100 rounds of robot tours in RSNs but can collect more data packets.

Figure 3

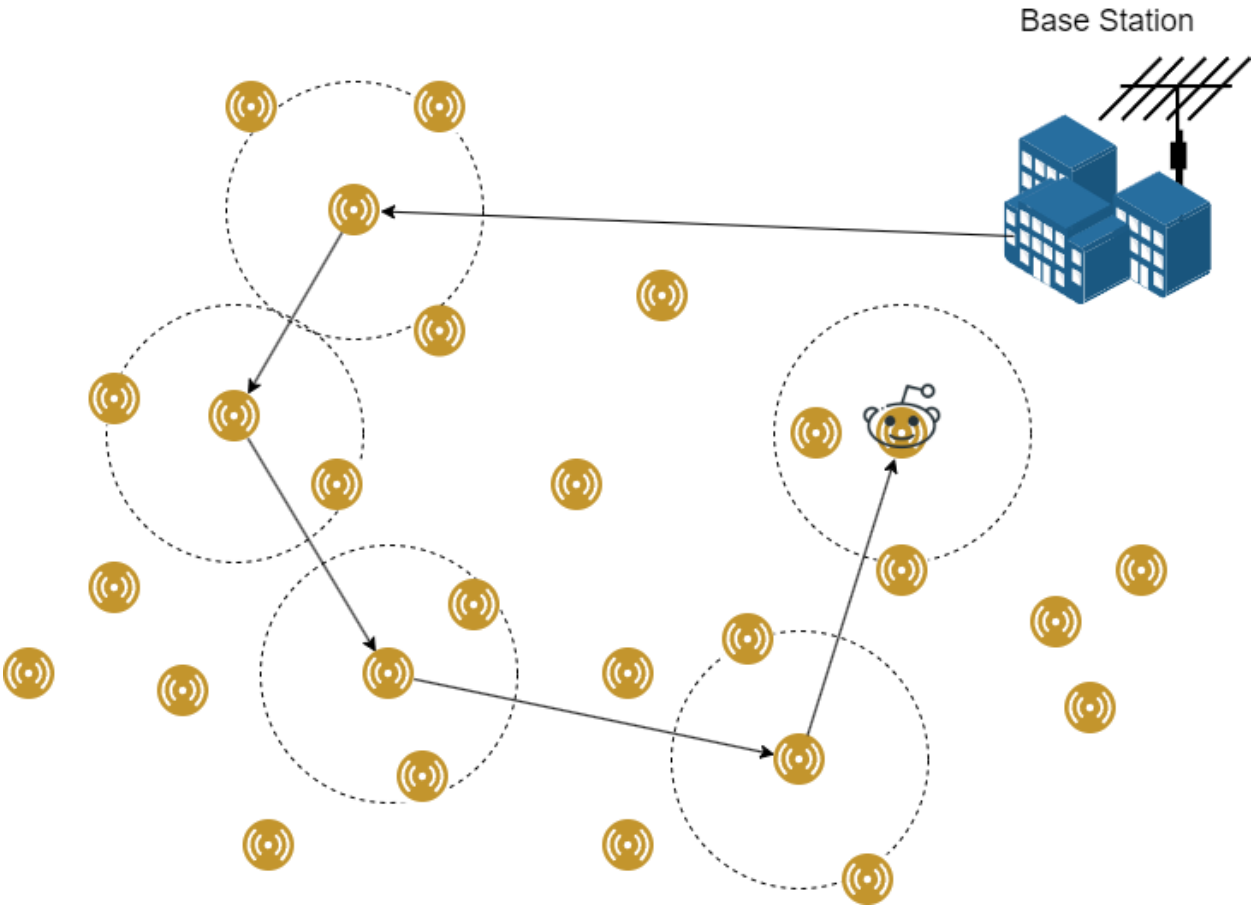
Robotic Sensor Network



Paper Organization. The rest of the paper is organized as follows. Chapter 2 reviews all the related work. Chapter 3 formulates the BC-TSP. Chapters 4 and 5 propose a suite of combinatorial algorithms, including an ILP optimal solution, three greedy heuristic algorithms and our MARL algorithm for the DCR. Chapter 6 compares our algorithms with the existing research and discusses the results. Chapter 7 concludes the paper with a discussion of future works.

Figure 4

Robotic Sensor Network BC-CSP Approach



CHAPTER 2

RELATED WORK

Robotic Sensor Networks (RSNs) (Gu et al., 2016; Huang et al., 2019) have garnered significant attention in recent years, with mobile robots being employed to enhance the performance of wireless sensor networks (Kim et al., 2017; Liu et al., 2020; Guo et al., 2014; Salarian et al., 2014; Wang & Chen, 2019; Kim et al., 2014; Liu et al., 2021; Xue et al., 2019).

Luo et al. (2005) identified a common issue in wireless sensor networks, where nodes near a base station bear the brunt of relaying data for a substantial portion of the network, leading to rapid battery depletion. They were among the pioneers in introducing robot mobility to mitigate this issue. Their proposed data-gathering scheme aims to minimize the maximum average load of sensor nodes by addressing both robot movement planning and data-gathering routing. Their approach assumes a circular sensor field and predominantly utilizes geometric calculations and techniques to minimize the maximum load on sensor nodes.

Ma et al. (2013) approached data-gathering in sensor networks from a graph-theoretical perspective, framing it as a covering salesman problem (CSP). The objective of the CSP is to minimize the length of the data-gathering tour undertaken by polling points visited by the robot. These polling points cover all sensor nodes within the network. Essentially, sensor nodes within the transmission range of a polling point directly transmit their data packets to that polling point, which are then collected by the robot. In scenarios involving multiple robots, the aim is to minimize the number of mobile robots while ensuring that each CSP subtour adheres to a specified time constraint. Ma et al. formulated these problems as mixed-integer programs and devised heuristic data-gathering algorithms. Building upon this work, Guo et al. (2014) extended the approach by incorporating wireless energy-charging into mobile data collection. They

formulated a network utility maximization problem that accounts for energy balance and the limited time each mobile collector can spend at a particular location. Additionally, they devised a distributed algorithm to address these challenges.

Salarian et al. (2014) expanded on the concept of polling points (PPs) by introducing rendezvous points (RPs) into the data-collection process. In this framework, a hybrid moving pattern is formed where a mobile-sink node exclusively visits RPs, while sensor nodes that are not RPs relay their sensed data via multi-hop communication to the nearest RP. Unlike PPs, which facilitate one-hop communication between sensor nodes and RPs, RPs serve as central points for data aggregation. Salarian et al. devised a weighted rendezvous planning heuristic algorithm to enable a mobile sink to retrieve all sensed data within a specified deadline while minimizing the energy expenditure of sensor nodes. In a related study, Wang et al. (2016) noted that the sensors produce data at the same rate and do not face limitations on buffer size in the approach. To address these assumptions, they relaxed these constraints and developed efficient path planning for a reliable data-gathering algorithm.

The problems discussed above assume active participation of sensor nodes in the data collection process, typically by transmitting their data packets to polling points or rendezvous points, as seen in the covering salesman problem. However, this approach can rapidly deplete sensor node battery power due to the energy-intensive nature of wireless communication, potentially rendering the entire RSN inoperable. In contrast, we propose a budget-constrained traveling salesman problem, where the robot (representing the traveling salesman) directly visits each sensor node to collect its data packet. In this scenario, since the robot interacts directly with sensor nodes, the distance between them can be assumed to be zero, effectively minimizing

transmission energy consumption. We demonstrate that compared to the existing covering salesman approach, our method significantly prolongs the lifetime of the RSN.

In a related context, Kim et al. (2017, 2014) explored multiple-drone-assisted search-and-reconnaissance scenarios, framing them as the Travelling Salesman Problem with Neighborhood (TSPN). In TSPN, a node is considered visited once its distance from the traveler falls below a certain threshold value. The authors devised approximation algorithms to minimize task completion time and the largest time gap between consecutive observations of the same point of interest. Meanwhile, Liu et al. (2020) concentrated on mobile data collection in disconnected sensor networks, proposing a path-planning strategy to achieve full connectivity for partitioned WSNs and construct shorter paths.

All the aforementioned works assume that the mobile robots in the RSN possess sufficient battery power to accomplish the objectives outlined in those studies. However, in a large-scale sensor field, it's plausible that the robot lacks the necessary battery power to visit all sensor nodes. In scenarios where sensor nodes generate sensory data with varying priorities and values (i.e., prizes), a critical question arises: how to schedule the robot to visit and collect data from nodes with maximum prizes before returning to the charging station for recharging.

Other groups of research consider the existence of physical resource constraints in the RSN or impose time constraints on data packet collection. Guo et al. (2014, 2016) proposed a framework for joint wireless energy replenishment and mobile data gathering, presenting a network utility maximization problem constrained by flow, energy balance, link and battery capacity, and the limited sojourn time of the mobile collector. They introduced a suite of sub algorithms to achieve cross-layer data control, scheduling, and routing for sensor nodes, along with sojourn time allocation for the mobile collector at different anchor points. However, this

framework still adopts a CSP approach, which may not be energy-preserving for the sensor nodes.

Chen et al. (2016) identified scenarios in data harvesting where certain data are time-sensitive and may become outdated after a period. Consequently, they introduced a time-constrained data harvesting problem aimed at determining an optimal data collection path within an RSN to maximize data collection within a specified time frame, offering a constant-factor approximation algorithm. Their consideration of time constraints in data collection aligns with the battery power constraint addressed in this paper. However, their robot mobility model differs fundamentally from ours. They assume the robot can collect data from any nodes within a constant distance along its travel path, while our model necessitates the robot to reach each sensor node to collect its packets directly. Additionally, they assume each sensor node contains one unit of data message, aiming to cover as many sensors as possible. In contrast, our model accounts for varying numbers of data packets per sensor, focusing on maximizing the collection of data packets.

CHAPTER 3

PROBLEM FORMULATION

Network Model. We model the RSN as a rectangular area of l meters by m meters, in which n sensor nodes and one robot is located. Let $V_s = \{1, 2, ..., n\}$ denote the set of sensor nodes randomly located in the RSN, with $i \in V_s$ located at the location (x_i, y_i) where $0 \leq x_i \leq l$ and $0 \leq y_i \leq m$. Each sensor node $i \in V_s$ has $d_i \geq 0$ number of data packets; each having a size of k bits. Let $r = (0, 0)$ denote the depot of the RSN, where the robot and a base station are located. The robot is dispatched from the depot to collect data from the RSN and then returns to the depot to upload its collected data to a base station. The depot is also a charging station where the battery of the robot can be recharged.

Let $V = Vs \cup \{r\}$ and $\forall i, j \in V$, let $d(i,j) = \sqrt{(xi - xj)^2 + (yi - yj^2}$ denote the distance between any pair of sensor nodes i and j or a sensor node i and the depot $r = j$. We assume both robot and sensor nodes have the same transmission range T_r; i.e., when the distance between a sensor node and the robot is within T_r, the robot can collect the sensor node's data packet wirelessly. We leave the more general case that the robot can adjust its transmission range for a more energy-efficient data collection as a future work.

Energy Model for Data Collection. We use the well-known first-order radio wireless model (Heinzelman et al., 2000) to quantify the energy consumption when collecting data from sensors by robots. When sensor node i sends a k-bit data packet to the robot r over their distance l_i, $r \leq T_r$ meters, the *transmission energy* spent by i is $E_i^t(r) = \epsilon_{elec} * k + \epsilon_{amp} * k * l_{i,r}^2$, the receiving energy spent by r is $E_v^{re} = \epsilon_{elec} * k$. Here $\epsilon_{elec} = 100nJ/bit$ is the energy consumption per bit on the transmitter circuit and receiver circuit, and $e_{amp} = \frac{100pJ}{bit}/m^2$ is the energy consumption per bit on the transmit amplifier.

Mobility Model of the Robot. Next, we present the mobility model for the robot; i.e., how it moves around the RSN and collects data packets from the sensor nodes. Gu et al. divides the mobility management and control of mobile sinks (i.e., robots) in the RSN into four categories: uncontrollable mobility (UMM), path-restricted mobility (PRM), location-restricted mobility (LRM), and unrestricted mobility (URM). UMM characterizes the uncontrollable behavior of mobile sinks (e.g., sinks are attached to wild animals moving randomly). PRM models the geographic constraints of sinks on pre-defined paths (e.g., freeways and railways). LRM characterizes some scenarios wherein mobile sinks can only stop at certain locations to collect data due to geographic and structural constraints (e.g., structural health monitoring). URM models that the mobile sink has total freedom of mobility. Simpler than PRM, where the trajectory of the mobile sink is pre-determined, or LRM, where the stops are fixed, URM is widely used in the existing literature to study both data routing and robot motion control in the RSN.

We thus adopt the URM model and assume the robot can stop at any location in the RSN to collect data packets in this paper. In particular, to collect data from sensor node *i*, the robot moves from its current location to sensor node i's location (x_i , y_i) following the straight line between them. At (x_i , y_i), the robot collects the data packets from sensor node *i* as well as all the sensor nodes *j* where $d(i, j) \leq T_r$. That is, when the robot is located at (x_i , y_i), the maximum amount of data packet it can collect is sum of all data packets in the range. And denote it as p_i. The goal of the robot is to find a sequence of sensor nodes to stop and collect data packets such that it can collect a maximum number of data packets before it runs out of battery power and returns to the depot, where it uploads the collected data to the base station and recharge its battery.

Table 1

Notation Summary

Notation	Description
V_s	The set of n sensor nodes
d_i	Number of data packets at sensor node $i \in V_s$
r	The depot where the robot is located and recharged
T_r	Transmission range of sensor nodes and robot
$\mathcal{B}$	The initial battery power of the robot
r	The depot of the RNS, where robot is stationed and recharged
$d(i,j)$	Distance between two nodes i and j in the RSN
E_i	Initial energy level of sensor node i
$E_i^{'}$	Remaining energy level of sensor node i after data offloading
$E_u^t(v)$	Transmission energy spent by u to transmit one packet to v
E_v^r	Receiving energy spent by v to receive one data packet
E_v^s	Storing energy spent by v to store one data packet
R	The data collecting route of the robot
B_R	The battery power of the robot used on this route R
P_R	The prizes collected by the robot on this route R
$\sigma(i,j)$	Node i's successor node in P_j
$y_{i,j}$	Node i's energy cost of offloading data packet D_j
$\xi(i)$	Number of data packets stored at storage node i

Energy Model of the Robot. Xiao et al. (2014) analyzed energy utilization in mobile robot traverse and estimates the maximum range achievable by wheeled mobile robots operating on a single battery discharge. It shows that a robot's energy consumption mainly includes two parts, viz., robotics and mobility. Robotics energy consumption is for computing, sensing, and communication while mobility consumption includes all the energy needed to keep the robot in motion, such as the drive motor, steering motor, and their related energy losses. In our case, according to our data collection model, the robot must arrive at the location of sensor nodes to collect data packets; thus, the distance between the robot and sensor nodes is considered zero. Thus, we assume that robotics energy is negligible and only focus on the energy consumption for robot mobility. We leave the integrated study of both robotics and mobility energy as a future

work. The energy required for mobility depends on the terrain type, traverse distance, and the robot's weight. Xiao et al. (2014) shows the ideally achievable distance *d* (in meters) of battery powered wheeled mobile robots on one single charge is

$$d = \frac{E}{w * C_{crr}}$$

where *E* (in joules) is the battery power of the robot, *w* (in Kg) is the weight of the robot, and C_{crr} represents the coefficient of rolling friction, which depends on the terrain type. Note that as with the increase of robot velocity, the robot mobility power increases while the robot traveling time decreases. Therefore, the final mobility energy is unchanged and thus the robot's mobility consumption doesn't depend on rover velocity or traveling time.

ILP Solution. We first solve the DCR optimally by formulating it as an integer program ILP(A). Decision variable $x_{i,j}$ indicates if edge (*i*, *j*) is on the prize-collecting route (i.e., node *j* is visited immediately after node *i* is visited); $x_{i,j}$ = 1 if so and 0 otherwise. We introduce $|V| - 1$ position variables u_i , i ∈ V − {s}, to indicate the order in which the nodes are visited. u_s = 1 as s is the starting node and $u_i < u_j$ indicates that node *i* is visited before node *j* (but not necessarily immediately). $u_i - 1$ equals the number of edges along the prize-collecting route when going from node *s* to node *i*. Objective function 1 is to maximize the total collected prizes. Constraint 3 is the integer constraint of $x_{i,j}$. Constraint 4 guarantees that the prize-collecting route starts at node *s* and ends at node *t*. Constraint 5 ensures the connectivity of the path and that each node is visited at most once. Constraint 6 guarantees that the total traveling cost on the path does not exceed the given budget of *B.*

Figure 5

Integer Linear Program Solution for DCR

$$(A) \quad \max \sum_{i \in V-\{t\}} \sum_{j \in V-\{s\}} p_i \cdot x_{i,j} \tag{2}$$

s.t.

$$x_{i,j} \in \{0,1\} \quad \forall i,j \in V \tag{3}$$

$$\sum_{j \in V-\{s\}} x_{s,j} = \sum_{i \in V-\{t\}} x_{i,t} = 1, \tag{4}$$

$$\sum_{i \in V-\{t\}} x_{i,k} = \sum_{j \in V-\{s\}} x_{k,j} \leq 1, \quad \forall k \in V - \{s,t\} \tag{5}$$

$$\sum_{i \in V-\{t\}} \sum_{j \in V-\{s\}} w_{i,j} \cdot x_{i,j} \leq \mathcal{B}, \tag{6}$$

$$2 \leq u_i \leq |V|, \quad \forall i \in V - \{s\} \tag{7}$$

$$u_i - u_j + 1 \leq (|V| - 1) \cdot (1 - x_{i,j}), \tag{8}$$

Constraints 7 and 8 combined are called Miller–Tucker–Zemlin (MTZ) Subtour Elimination Constraints [1]. They guarantee that there is one global tour visiting all the selected vertices instead of multiple subtours each visiting only a subset of the selected vertices.

CHAPTER 4

COMBINATORIAL ALGORITHMS

Below, we design two greedy heuristic algorithms viz. Algo. 1 and 2. We also use the Minimum Spanning Tree algorithm to compare our results. We first give the below definition.

Definition 1: (Budget-Feasible Nodes.) Given the current node *r* the traveling salesman is located and his available budget *B*, the budget-feasible nodes, denoted as $\mathcal{F}(r, B)$, is *s*'s unvisited neighbor nodes that the salesman can travel to and then return to destination node *t* with enough budget. That is, $\mathcal{F}(r, B) = \{u \mid (r, u) \in E \ \wedge (w(r, u) + w(u, t) \leq B) \wedge u \in U\}$[1] where *U* is the set of unvisited nodes.

Greedy Algorithm 1. In Algo. 1, at any node, the salesman always visits a budget-feasible node with the largest prize. It first sorts all the nodes in the descending order of their prizes (line 2) and then takes place in rounds (lines 4-12). In each round, with the current node *r* and the currently available budget *B*, it checks if there still exists unvisited and budget feasible nodes (line 4). If so, it visits the one with the largest available prize and updates all the information accordingly (lines 5-10). It stops when there are no unvisited nodes, or all the unvisited nodes are not budget-feasible (line 4), at which it goes to the destination node *t* and returns the route with its total cost, total prizes collected, and its remaining budget (lines 13 and 14). Its time complexity is O ($|V|^2$). Algo. 1 also works for the problem where *s* = *t*. Algorithm 1 is given in the following Figure 6.

Greedy Algorithm 2. Given an edge *(u, v)* $\in E$, and the traveling salesman is at node *u*, we define the prize cost ratio of visiting *v*, denoted as *pcr(u, v)*, as the ratio between the prize available at *v* and the edge weight *w(u, v)*. That is, *pcr(u, v)* = p_v / *w(u, v)*. Algo. 2 is similar to

Algo. 1, except it visits a budget-feasible node with the largest prize cost ratio in each round. Its time complexity is O ($|V|^2$).

Figure 6

Greedy Algorithm 1

Input: A weighted graph $G(V, E)$, s, t, and initial budget $\mathcal{B}$.
Output: A route R from s to t, its cost C_R and prize P_R.
Notations: R: the current route found, initially $\{s\}$;
C_R: the length (i.e., the cost) of R, initially zero;
P_R: the prizes collected on R, initially zero;
U: the set of unvisited nodes, initially $V - \{s, t\}$;
r: the current node where the salesman is located;
$c(r, t)$: the cost of the shortest path between r and t;
B: current available budget, is $\mathcal{B}$ initially;

1. $r = s$, $R = \{s\}$, $C_R = P_R = 0$, $B = \mathcal{B}$,
   $U = V - \{s, t\} = \{v_1, v_2, ..., v_{|V|-2}\}$;
2. Sort nodes in U in descending order of their prizes;
   WLOG, let $p_{v_1} \geq p_{v_2} ..., \geq p_{v_{|V|-2}}$;
3. $k = 1$; // the index of the node with largest prize
4. **while** $(U \neq \phi \wedge \mathcal{F}(r, B) \neq \phi)$
5. **if** $(v_k \in \mathcal{F}(r, B))$
6. $R = R \cup \{v_k\}$;
7. $C_R = C_R + w(r, v_k)$, $P_R = P_R + p_{v_k}$;
8. $B = B - w(r, v_k)$, $U = U - \{v_k\}$;
9. $r = v_k$;
10. **end if;**
11. $k++$;
12. **end while;**
13. $R = R \cup \{t\}$, $C_R = C_R + c(r, t)$, $B = B - c(r, t)$;
14. **RETURN** R, C_R, P_R, B.

Figure 7

Greedy Algorithm 2

Input: A complete weighted graph $G(V, E)$, s, t, and $\mathcal{B}$.
Output: A route R from s to t, its cost C_R and prize P_R.
Notations: R: the current route found, starts from s;
C_R: the length (i.e., the cost) of R, initially zero;
P_R: the prizes collected on R, initially zero;
U: the set of unvisited nodes, initially $U = V - \{s, t\}$;
r: the node where the salesman is located currently;
B: current remaining budget, is $\mathcal{B}$ initially;

1. $r = s$, $R = \{s\}$, $C_R = P_R = 0$, $U = V - \{s, t\}$;
2. $B = \mathcal{B}$;
   // if not all nodes are visited, and there are feasible nodes
3. **while** $(U \neq \phi \wedge \mathcal{F}(r, B) \neq \phi)$
4. Let $u = \text{argmax}_{v \in \mathcal{F}(r,B) \cap U} pcr(r, v)$;
5. $R = R \cup \{u\}$;
6. $C_R = C_R + w(r, u)$, $P_R = P_R + p_u$;
7. $B = B - w(r, u)$, $U = U - \{u\}$;
8. $r = u$;
9. **end while;**
10. $R = R \cup \{t\}$, $C_R = C_R + w(r, t)$, $B = B - w(r, t)$;
11. **RETURN** R, C_R, P_R, B.

Spanning Tree Covering Algorithm (Yang et al., (2013)) The fundamental concept behind their greedy algorithm involves selecting a subset of points from the candidate polling point set, where each point corresponds to a neighbor set of sensors. At each step of the algorithm, a neighbor set of sensors can be covered by selecting its corresponding candidate polling point as a polling point in the data gathering tour. The algorithm continues until all sensors are covered, at which point it terminates. In our paper we refer to this algorithm as Yang's. For our

implementation we have updated the algorithm's next node selection. In Yang's the next node is found by minimizing

$$\alpha = \frac{cost\{S\}}{|S \cap U|}$$

Where *S* is the neighbor set of a node *n.* Their approach focuses on minimizing cost by considering the number of nodes in the neighbor set, whereas ours aims to maximize the total data packets across all sensor nodes by weighing the cost to reach them. This fundamental disparity results in the collection of a greater number of data packets in our approach.

Figure 8. (Yang et al. (2013))

Yang's Algorithm

Create an empty set P_{curr}
Create a set U_{curr} containing all sensors
Create a set L containing all candidate polling points
while $U_{curr} \neq \Phi$
  Find a polling point $l \in L$, which minimizes $\alpha = \frac{cost\{nb(l)\}}{|nb(l) \bigcap U_{curr}|}$
  Cover sensors in $nb(l)$
  Add the corresponding polling point of $nb(l)$ into P_{curr}
  Remove the corresponding polling point of $nb(l)$ from L
  Remove sensors in $nb(l)$ from U_{curr}
end while
Find an approximate shortest tour on polling points in P_{curr}

For all these algorithms we are going to compare the results for the BC-TSP and BC-CSP approach to show that the latter collects more data packets given the budget. The only difference in our implementation of BC-CSP and BC-TSP is that the robot will have a transmission range to collect data packets from neighbors in BC-CSP. For the Covering Problem we use all the data

packets from the node within the range and in normal TSP it will just collect it from the node which it is visiting.

CHAPTER 5

MULTI-AGENT REINFORCEMENT LEARNING ALOGRITHM

Reinforcement Learning (RL). We describe an agent's decision-making in an RL system as a Markov decision process (MDP), which is represented by a 4-tuple *(S, A, t, r)*:

• *S* is a finite set of states,

• *A* is a finite set of actions,

• $t : S \times A \rightarrow S$ is a state transition function, and

• $r : S \times A \rightarrow R$ is a reward function, where R is a real value reward (Sutton & Barto, 2020).

In MDP, an agent learns an optimal policy that maximizes its accumulated reward. At a specific state $s \in S$, the agent takes action $a \in A$ to transition to state $t(s, a) \in S$ while receiving a reward $r(s, a) \in R$. The agent maintains a policy $\pi(s) : S \rightarrow A$ that maps its current state $s \in S$ into the desirable action $a \in A$. In the context of the BC-TSP, the states are all the nodes V , and the actions available for an agent at a node are all the edges emanating from this node. We consider a deterministic policy wherein, given the state, the policy outputs a specific action for the agent. A deterministic policy suits the BC-TSP well, as in BC-TSP, when an agent at a node takes action (i.e., follows one of its edges), it will surely end up with the node on the other end of the edge. A widely used class of RL algorithms is value-based (Sutton & Barto, 2020; Littman, 2001), which finds the optimal policy based on the value function at each state *s*,

$$V_s^{\pi} = E\{\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)) | s_0 = s\}.$$

The value at each state is the expected value of a discounted future reward sum with the policy π at state *s*. Here, γ ($1 \leq \gamma \leq 1$) is the discounted rate that determines the importance of

future rewards; the larger of the γ, the more important the future rewards. Recall that *r(s, π(s))* is the reward received by the agent at state *s* by taking action following policy *π*.

Q-Learning. Q-learning is a family of value-based algorithms (Sutton & Barto, 2020). It learns how to optimize the quality of the actions in terms of the Q-value *Q(s, a)*. *Q(s, a)* is defined as the expected discounted sum of future rewards obtained by taking action a from state *s* following an optimal policy. The optimal action at any state is the action that gives the maximum Q value. For an agent at state *s*, when it takes action *a* and transitions to the next state *t*, *Q(s, a*) is updated as

$$\mathrm{Q(s,a)} \leftarrow (1-\alpha)\cdot \mathrm{Q(s,a)} + \alpha \cdot [\mathrm{r(s,a)} + \gamma \cdot \mathrm{maxbQ(t,b)}], \quad (9)$$

where $1 \leq \alpha \leq 1$ is the learning rate that decides to what extent newly acquired information overrides old information in the learning process. In Eqn. 9, $\max_b Q(t, b)$ is the maximum reward that can be obtained from the next state *t*.

Multi-agent Reinforcement Learning (MARL) Algorithm. In our MARL framework for BC-TSP, there are multiple agents that all start from the node s. They work synchronously and cooperatively to learn the state-action Q-table and the reward table and act accordingly. We first introduce the action rules for all the learning agents and then present our MARL algorithm.

Action Rule of Agents. Each agent follows the same action rule specifying the next node it moves to during the learning process. It consists of the following three scenarios.

- **Exploitation.** In exploitation, the agent always chooses the node,

$$t = \ argmax_{u \in U \cap \mathcal{F}(s,B)} \left\{ \frac{[Q(s,u)]^{\delta} \ * \ p_u}{[w(s,u)]^{\beta}} \right\}$$

  to move to. Here, *U* is the set of nodes not visited yet by the agent and *F(s, B)* is node *s*'s budget-feasible nodes, and δ and β are preset parameters. That is, an agent, located at

node *s*, always moves to an unvisited and feasible node *u* that maximizes the learned Q-value *Q(s, u)* weighted by the length *w(s, u)* and the prize p_u available at node *u*. When $q \leq q_0$, where *q* is a random value in [0, 1] and q_0 ($0 \leq q_0 \leq 1$) is a preset value, exploitation is selected; otherwise, the agent chooses exploration explained below

- **Exploration.** In exploration, the agent chooses a node $t \in U \cap F(s, B)$ to move to by the following distribution:

$$p(s,t) = \frac{([Q(s,t)]^{\delta} * p_u\ )/[w(s,t)]^{\beta}}{\sum_{u \in U \cap \mathcal{F}(s,B)} ([Q(s,u)]^{\delta} * p_u\ )/[w(s,u)]^{\beta}}$$

  That is, a node $u \in U \cap F(s, B)$ is selected with probability *p(s, u)*, while $\sum_{u \in U \cap \mathcal{F}(s,B)} \mathrm{p(s,u)} = 1$. The distribution *p(s, t)* characterizes how good the nodes are at learned Q-values, the edge lengths, and the node prizes. The higher the Q-value, the shorter the edge length, and the larger the node prize, the more desirable the node is to move to.

- **Termination**. When an agent is located at node *s* and $U \cap F(s, B) = \varphi$, it does not have an unvisited budget feasible node. In this case, the agent goes to destination *t* and terminates in this episode.

MARL Algorithm. Next, we present our MARL algorithm viz. Algo. 3, which consists of a learning stage for the *m* agents (lines 1-32) and an execution stage for the traveling salesman (lines 33-39). The learning stage takes place in a preset number of episodes. Each episode consists of the two steps below. In the first step (lines 3-26), all the *m* agents are initially located at the starting node *s* with zero collected prizes. Then each independently follows the action rule to move to the next budget-feasible node to collect prizes and collaboratively updates the Q-value of the involved edges. This takes place in parallel for all the agents. When an agent can no

longer find a feasible unvisited node to move to due to its insufficient budget, it terminates and goes to t (lines 8-14); in this case, it must wait for other agents to finish in this episode. Otherwise, it moves to the next node, collects the prize, and continues the prize-collecting process (lines 15-23). In either case, it updates the Q-values of the involved edge. Here, we assume the prizes at each node can be collected multiple times (as this is the learning stage). In the second step (lines 27-31), the *m* agents communicate with each other and find among the m routes the one with the maximum collected prizes. It then updates the reward value and Q-value of the edges of this route. Finally, in the execution stage (lines 33-38), the traveling salesman starts from *s*, visits the node with the largest Q-value in the Q-table, and ends at *t*, collecting the prizes along the way. Note we set the initial Q-value and reward value for edge *(u, v)* as $\frac{p_u + p_v}{w(u,v)}$ and $\frac{-w(u,v)}{p_u}$, respectively, to reflect the fact that the more prizes available and less length of an edge, the more valuable of the edge for the salesman to travel. Algorithm 3 is divided into 3 parts below for better understanding. Figure 9 are the definitions and initial values. Figure 10 is the learning stage of the algorithm and Figure 11 shows the execution stage.

Figure 9.

Initialization of MARL

Algorithm 3:

MARL Algorithm for PC-TSP.

Input: A graph $G(V, E)$, s, t, and a budget $\mathcal{B}$.
Output: A route R from s to t, C_R, and P_R.
Notations: i: index for episodes; j: index for agents;
U_j: set of nodes agent j not yet visits, initially $V - \{s, t\}$;
R_j: the route taken by agent j, initially empty;
B_j: the currently available budget of agent j, initially $\mathcal{B}$;
l_j: the cost (i.e., the sum of edge weights) of R_j, initially 0;
P_j: the prizes collected on R_j, initially 0;
r_j: the node where agent j is located currently;
s_j: the node where agent j moves to next;
$isDone_j$: agent j has finished in this episode, initially false;
R: the final route found the MARL, initially empty;
$Q(u, v)$: Q-value of edge (u, v), initially $\frac{p_u + p_v}{w(u,v)}$;
$r(u, v)$: Reward of edge (u, v), initially $\frac{-w(u,v)}{p_v}$;
α: learning rate, $\alpha = 0.1$;
γ: discount factor, $\gamma = 0.3$;
q_0: trade-off between exploration and exploitation, $q_0 = 0.5$;
δ, β: parameters in node selection rule; $\delta = 1$ and $\beta = 2$;
W: a constant value of 100;
epi: number of episodes in the MARL, $epi = 30000$;

Figure 10

Learning Stage of MARL

1. **for** $(1 \leq i \leq epi)$ // Learning stage
2. All the m agents are at node s, $r_j = s, 1 \leq j \leq m$;
3. **for** $(j = 1; j \leq m; j{+}{+})$ // Agent j
4. $P_j = 0$, $B_j = \mathcal{B}_j$, $isDone_j = false$;

end for;

// At least one agent has not finished in this episode

5. **while** $(\exists\ j,\ 1 \leq j \leq m,\ isDone_j == false)$
6. **for** $(j = 1; j \leq m; j{+}{+})$ // Agent j
7. **if** $(isDone_j == false)$ // Agent j has not finished
8. **if** $(U_j \cap \mathcal{F}(r_j, B_j) == \phi)$ // Agent j terminates
9. $isDone_j = true$;
10. $R_j = R_j \cup \{t\}$; // Agent j goes to t
11. $l_j = l_j + w(r_j, t)$, $B_j = B_j - w(r_j, t)$;
12. $Q(r_j, t) = (1 - \alpha) \cdot Q(r_j, t) +$
$\alpha \cdot \gamma \cdot \max_{z \in U_j \cap \mathcal{F}_{(t, B_j)}} Q(t, z)$;
13. $r_j = t$;
14. **end if;**
15. **else**
16. Finds the next node s_j following action rule;
17. $R_j = R_j \cup \{s_j\}$;
18. $l_j = l_j + w(r_j, s_j)$, $B_j = B_j - w(r_j, s_j)$;
19. $P_j = P_j + p_{s_j}$; // Collect prize
20. $Q(r_j, s_j) = (1 - \alpha) \cdot Q(r_j, s_j) +$
$\alpha \cdot \gamma \cdot \max_{z \in U_j \cap \mathcal{F}_{(s_j, B_j)}} Q(s_j, z)$;
21. $r_j = s_j$; // Move to node s_j;
22. $U_j = U_j - \{s_j\}$;
23. **end else;**
24. **end if;**

25. **end for;**
26. **end while;**
27. $j* = \text{argmax}_{1 \leq j \leq m} P_j$; // Route of largest prize
28. **for** (each edge $(u, v) \in R_{j*}$)
29. $r(u, v) = r(u, v) + \frac{W}{P_{j*}}$; // Reward value $r(u, v)$
30. $Q(u, v) \leftarrow (1 - \alpha) \cdot Q(u, v) +$
$\alpha \cdot [r(u, v) + \gamma \cdot \max_b Q(v, b)]$; // Update Q-value
31. **end for;**
32. **end for;** // End of each episode in learning stage

Figure 11.

Execution Stage of MARL

// Execution stage
33. $r = s$, $R = \{s\}$, $C_R = 0$, $P_R = 0$, $B = \mathcal{B}$;
34. **while** $(r! = t)$
35. $u = \text{argmax}_b Q(r, b)$;
36. $R = R \cup \{u\}, C_R = C_R + w(r, u), P_R = P_R + p_u,$
$B = B - w(r, u)$;
37. $r = u$;
38. **end while;**
39. **RETURN** R, C_R, P_R, B.

CHAPTER 6

PERFORMANCE EVALUATION

Experiment Setup. We write our own simulator in Java on Windows 11 with AMD Processor (AMD Ryzen 5 4000 Series 6-Core) and 24GB of DDR4 Memory. In the plots Algorithm 1 and Algorithm 2 are denoted as Greedy 1 and Greedy 2 respectively. The Minimum Spanning Tree is given as Yang's (Yang et al., (2013)) and the algorithm 3 is stated as MARL. We generate a random sensor network with a size of 10,000m in length and 10,000m in width. 100 Nodes are randomly generated in the given filed with a random number of data packets ranging between [0,100]. For the covering salesman problem, we have kept the wireless transmission range of the robot to be 500m. The budget for the robot is in Kilo Watt-Hours (KWh). The base station is at location 0,0 for all the runs.

Comparing Greedy 1, Greedy 2, Yang's and MARL. Figure 11. Shows the comparison of the three combinatorial algorithms and MARL. For this simulation a single instance of the network was run 10 times with varying budgets of 0.5, 1, 1.5, and 2 KWh. We observe that the MARL, Greedy 2 and Yang's all outperform the Greedy 1. This is because Greedy 1 always chooses the node with highest data packet and does not consider the distance, which costs in using up the budget too early and not collecting more data packets. Moreover, our Greedy 2 algorithm outperforms Yang's algorithm due to the difference in the function for calculating the next node. We use the ratio of number of data packets to the distance while they use only distance as a factor for the TSP. The MARL algorithm outperforms all the other algorithms in collecting data packets. The figure clearly demonstrates that MARL is more efficient prize wise than the combinatorial algorithms.

Figure 12

Data Packets Collected vs Budget in Combinatorial Algorithms and MARL

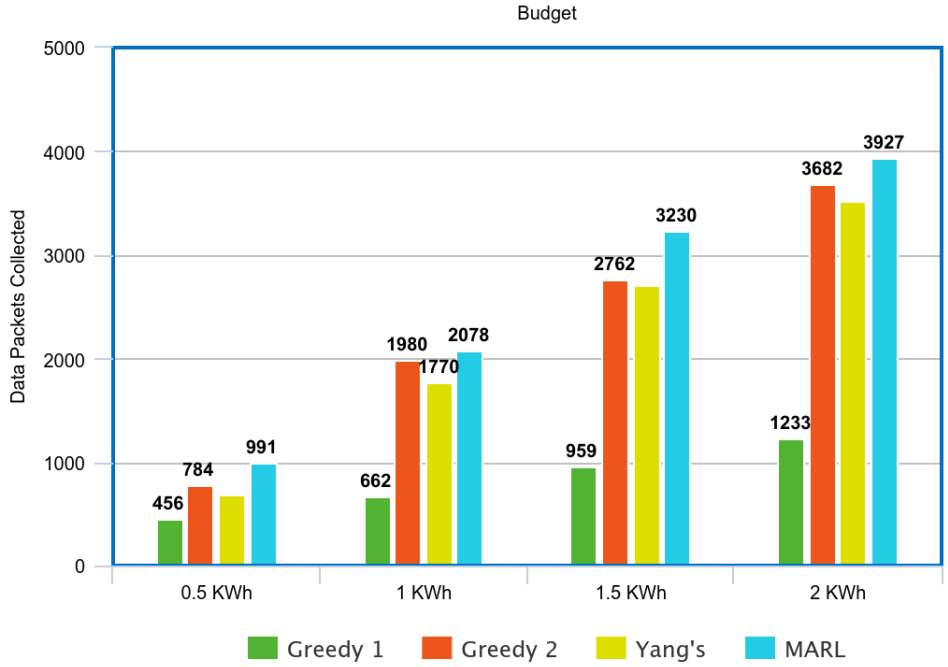


Figure 13. shows the budget all the algorithms use during the robot's one run through the sensor network. All the algorithms are built in a way that they can use up maximum of their budget and return home. Also given the robot can go to any node from the base station as it is not bounded by any transmission range, we see that all the algorithms use up almost the initial budget which is given to them.

Figures 14, 15,16 and 17 show the different routes taken by the robot for the same instance of a sensor network with a budget of 1KWh. Figure 14 depicts how less nodes are visited by Greedy 1 given its tendency to maximize the profit. Figures 15 and 16 are the Greedy 2 and Yang's Algorithms which visit almost same number of nodes, but their prominent difference is seen in Figure 12. Figure 17 shows the route taken by the robot performing MARL algorithm which is the best amongst these rest.

Figure 13

Budget Consumption vs Total Budget in Combinatorial Algorithms and MARL

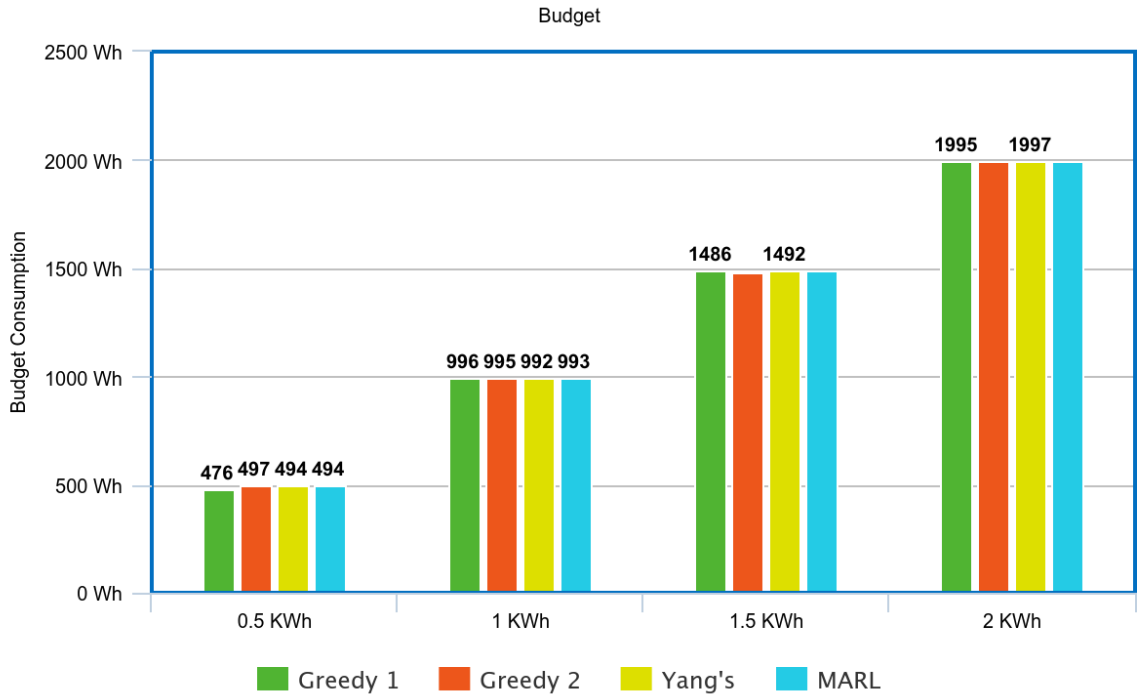


Figure 14

Route taken by Greedy 1

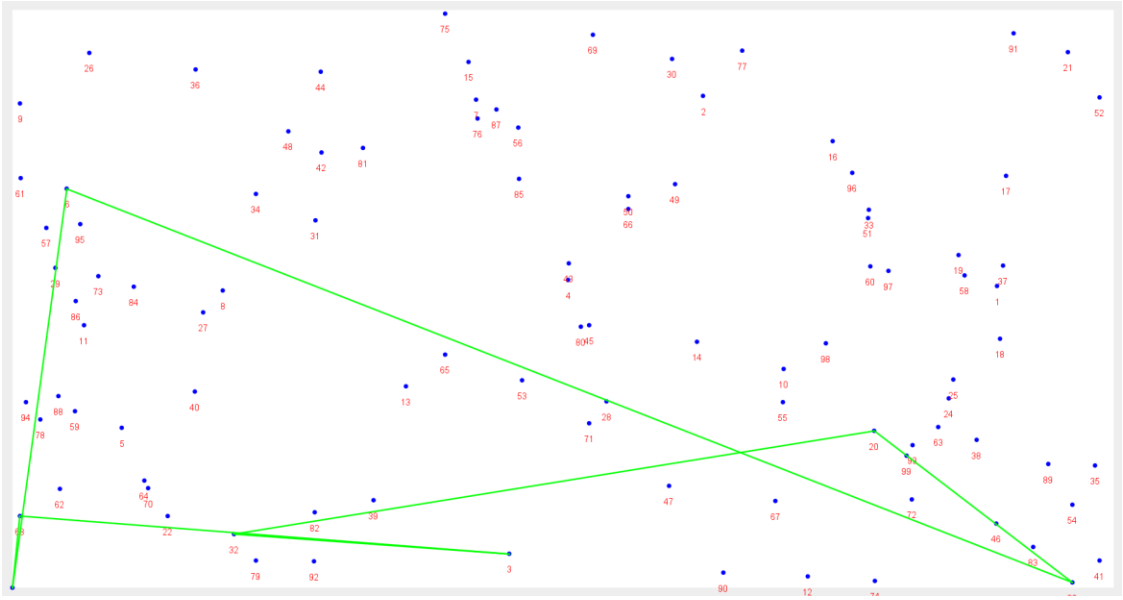


Figure 15

Route taken by Greedy 2

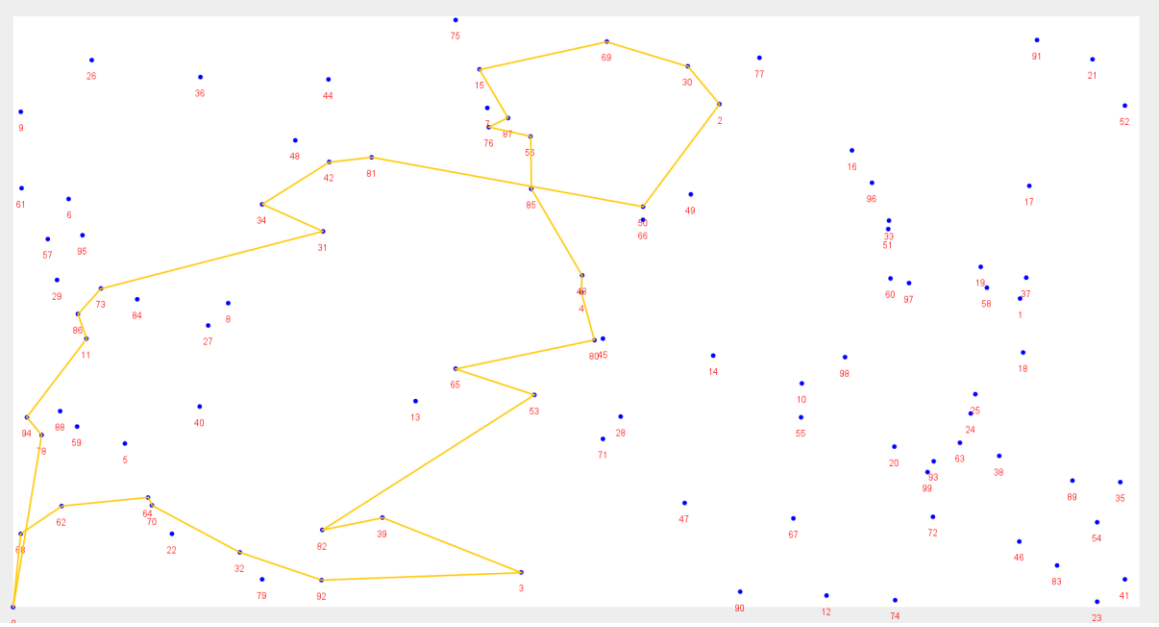


Figure 16

Route taken by Yang's.

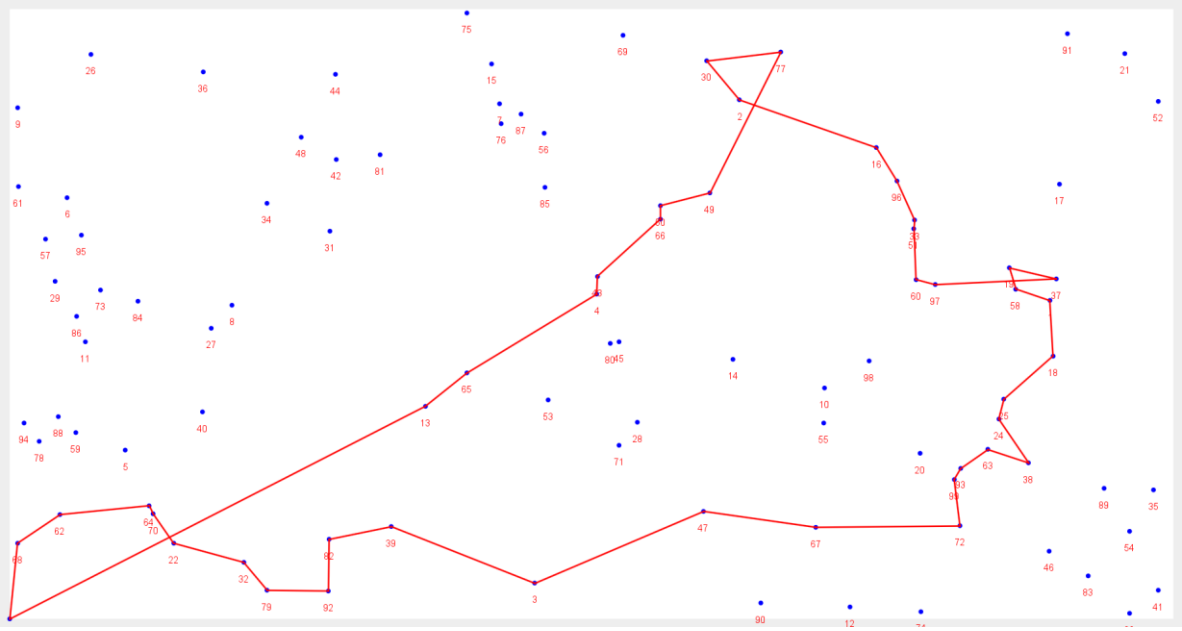
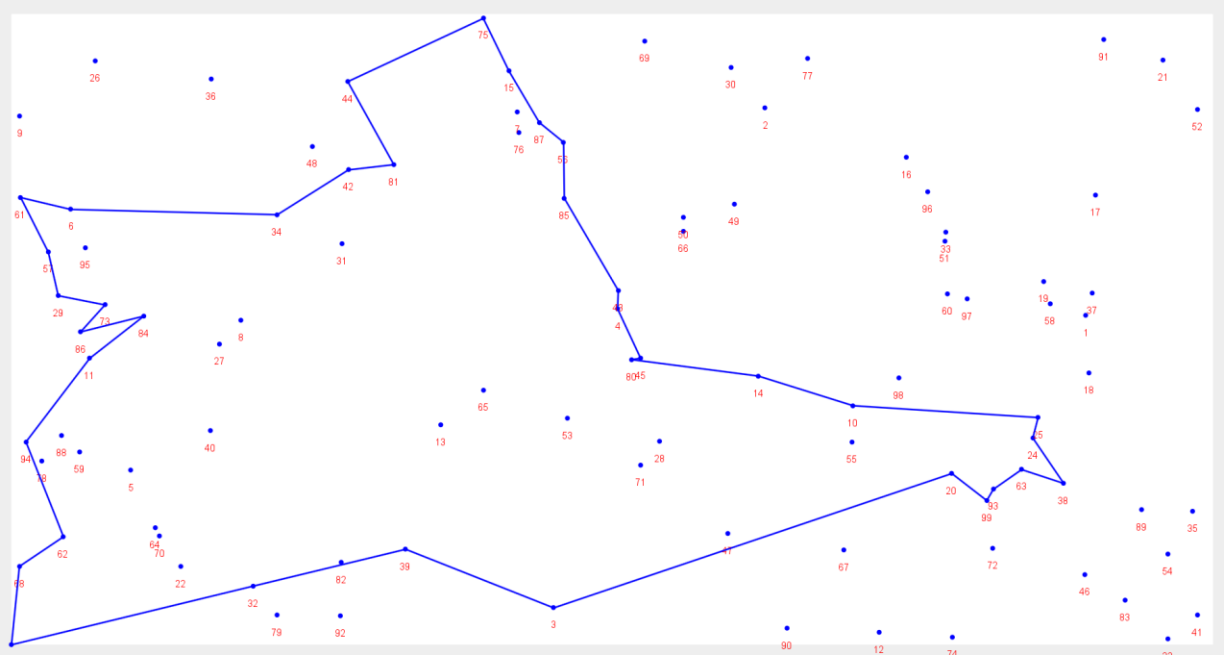


Figure 17

Route taken by MARL.



Impacts of Number of Agents *m* on MARL. Next, we study the impact of the number of agents *m* on the MARL's performance. We vary m from 1,5,10 to 15 and the budget from 0.5, 1, 1.5 to 2 KWh. Figure 18 shows that for each *m*, the higher the budget *B*, the larger the collected data packets. This is only true when there is not enough budget to collect all the available prizes. Figure 19 shows the traveled distance of the MARL with respect to *m* and *B*. The higher the *B*, the more distances it can travel. We observe that varying *m* does not seem to affect the traveled distance of the salesman. In Figure 20, as the number of agents *m* increases, the execution time of the MARL algorithm also increases for each budget *B*. This is attributed to the fact that the algorithm is configured to run a fixed number of episodes (10,000), and with more agents participating in the learning process, each episode takes longer to execute.

Figure 18

Data Collected vs Agents.

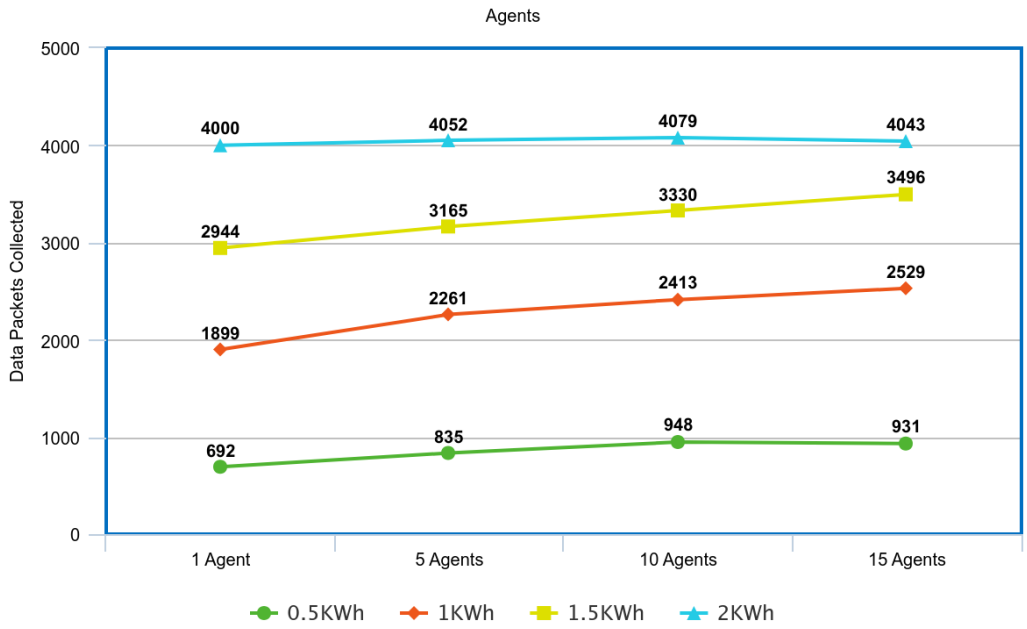


Figure 19

Total Distance Covered vs Agents

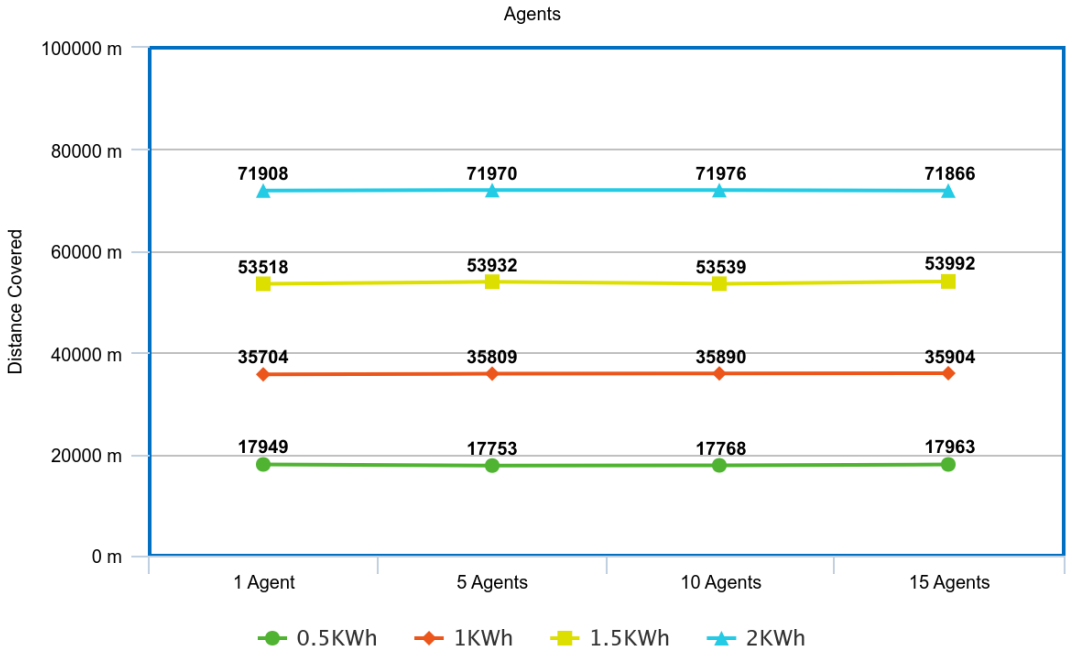
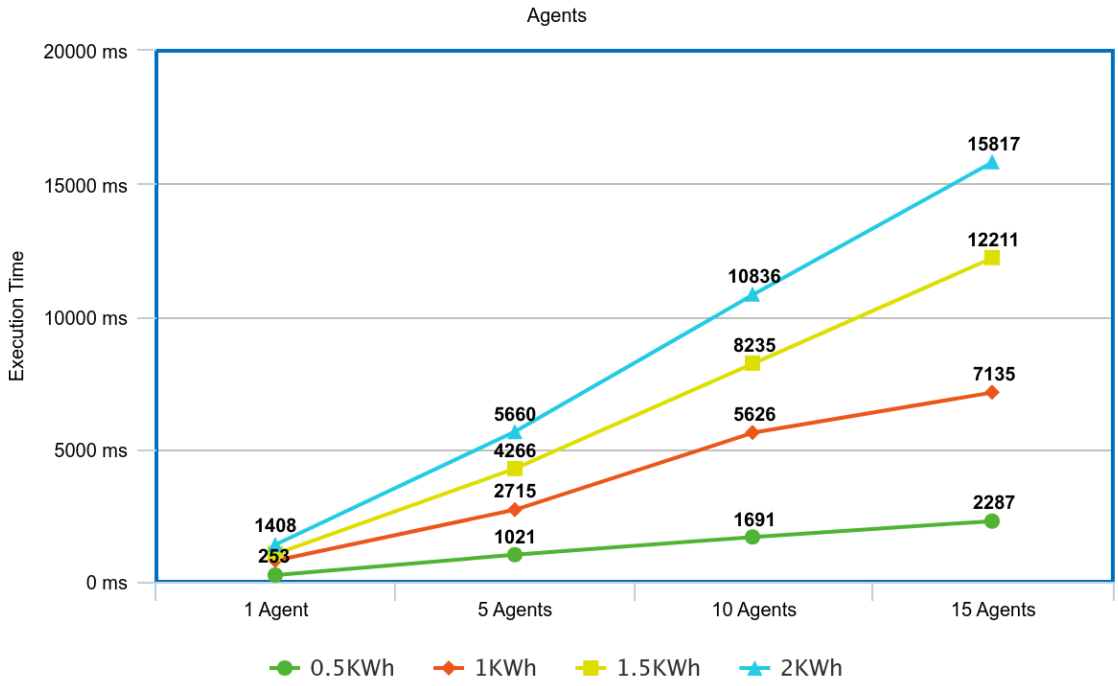


Figure 20

Execution Time vs Agents



Now we compare the MARL with the optimal ILP Solution. As ILP takes a long time to execute, we focus on a smaller sensor network of size 100m by 100m with 20 nodes in it. The start and end point for the robot is still the same i.e. base station (0,0). Figures 21 and 22 show the prize collected and distance travelled respectively, by varying the amount of initial budget from 50Wh to 120Wh. At 120Wh both can collect all the data packets from the sensor network. We observe that our MARL performs very similar to ILP in collecting data packets. The distance travelled is almost the same in both cases but a little higher when the budget is big in MARL. The average difference in the performance of MARL compared to ILP is 0.2%. This shows that MARL indeed is optimal for solving DCR.

Comparing Greedy Algorithms for BC-CSP. From our previous results we have established that MARL is optimal to solve the BC-TSP. We now move onward to the evaluation of greedy algorithms for the BC-CSP approach. We will be just focusing on the data collection part and not

on the battery consumption/usage of the sensor node as the goal here is to show the amount of data collected in limited battery capacity of the robot. Figure 23 shows the route taken and the sensor nodes covered by the greedy 1 algorithm for covering salesman approach. The robot collects data packets from the nodes within the range hence maximizing the collection in a single run. After comparing the routes from figures 23 and 14 we can conclude that CSP approach covers more nodes.

Figure 21

Data Packets Collected in MARL & ILP

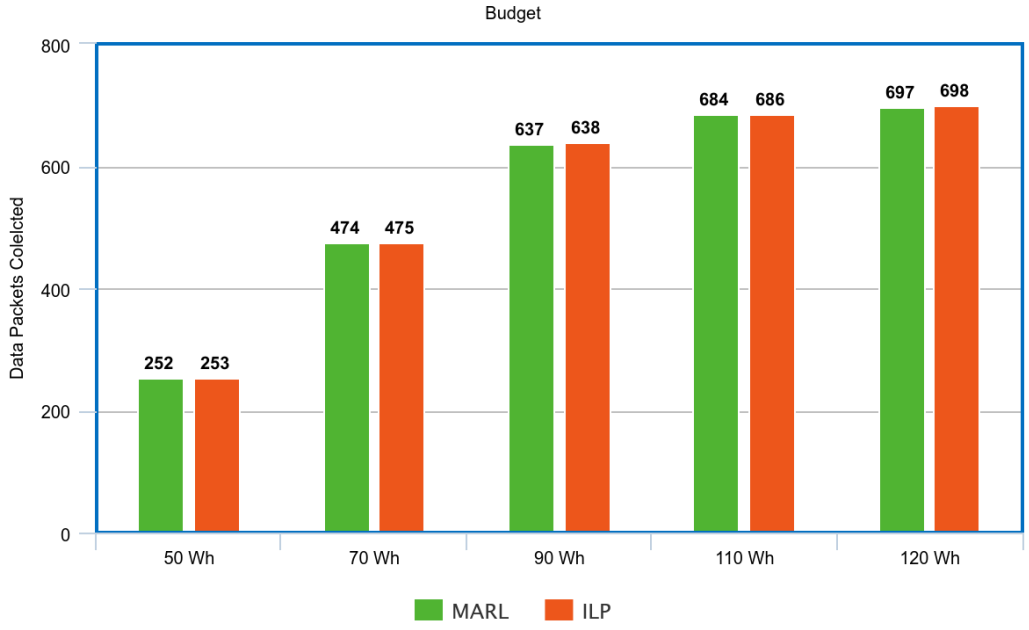


Figure 22

Distance Travelled in MARL & ILP

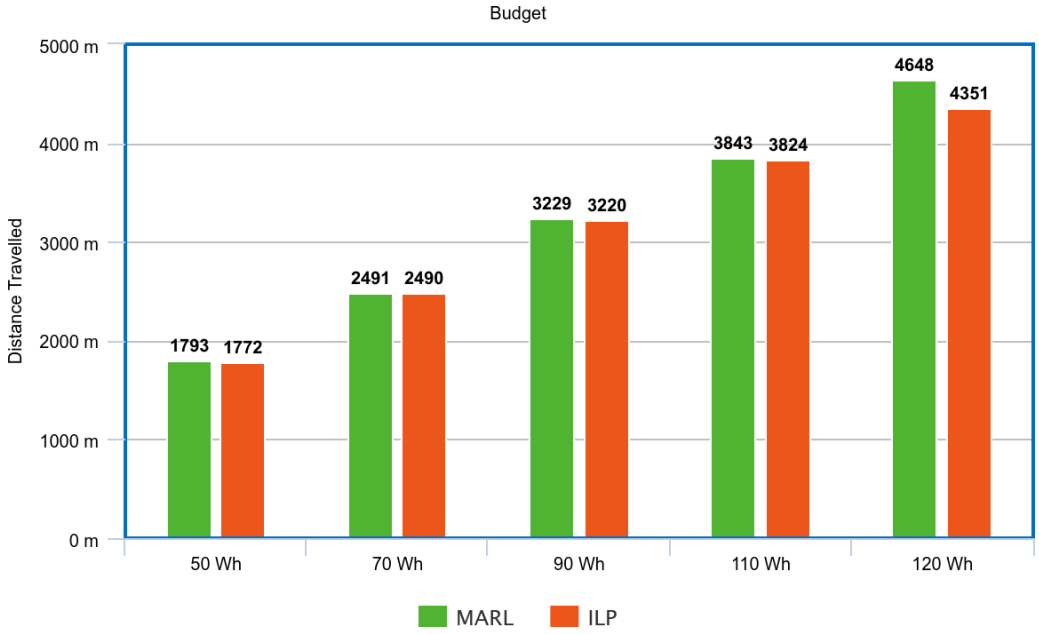


Figure 23

Route of Greedy 1 Algorithm (BC-CSP)

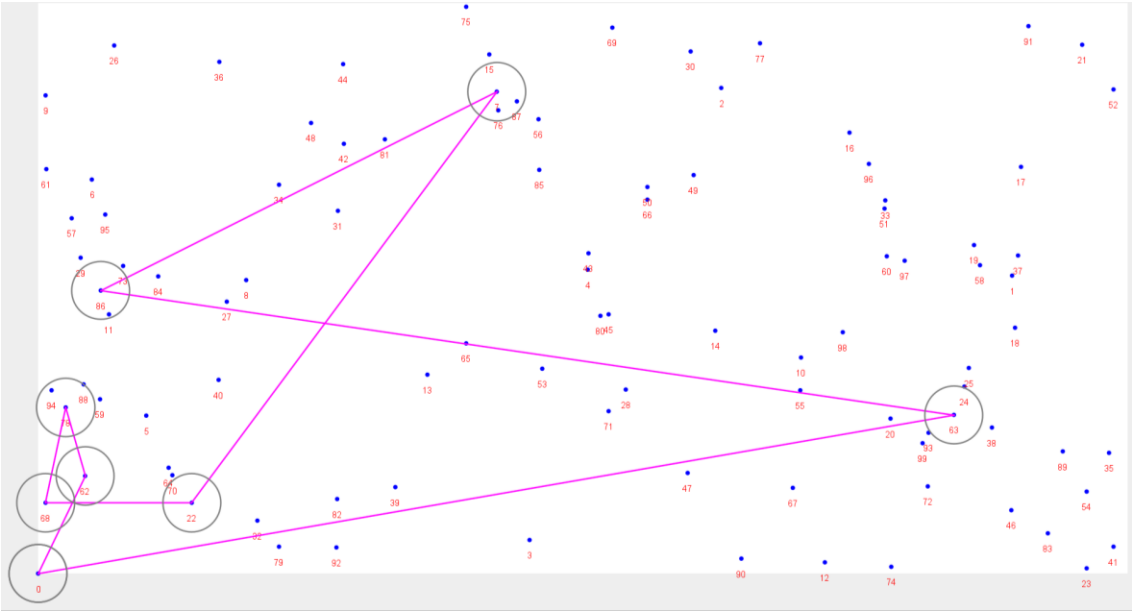
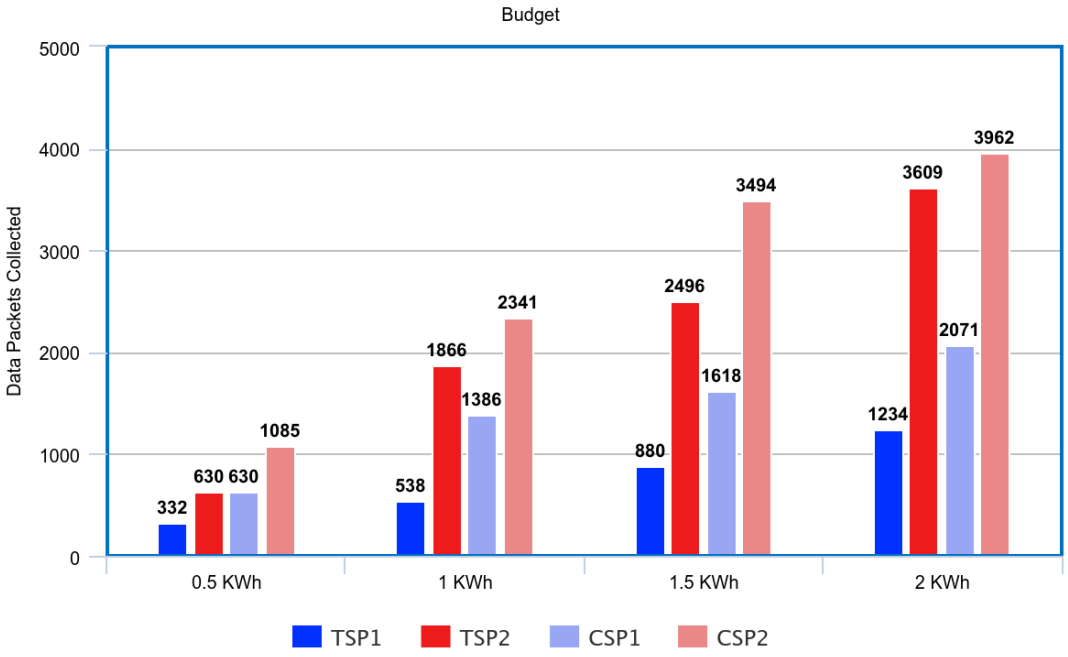


Figure 24

Data Packets Collected vs Budget (CSP)



In Figure 24 we can see that the lighter shades which represent the CSP algorithms collect more data packets than their counterparts in darker shade. An increase of 40% is seen in the collection. The distance travelled is also less given the data packets collected. This shows us that the BC-CSP approach can collect more data packets than BC-TSP approach.

Comparing Yang's Spanning Tree Covering Algorithm Figure 25 illustrates the comparison between the disparity of Yang's cost function and our own, as elaborated in Chapter 4. Our algorithm's cost function proves to gather a greater number of data packets compared to Yang's. This underscores the notion that while covering numerous nodes in a neighbor set, it's not guaranteed that they will contain more data packets. To enhance data packet collection, it's imperative to consider the data packets present at each sensor node.

Figure 25

Comparison of Yang's Cost Function and Our Approach

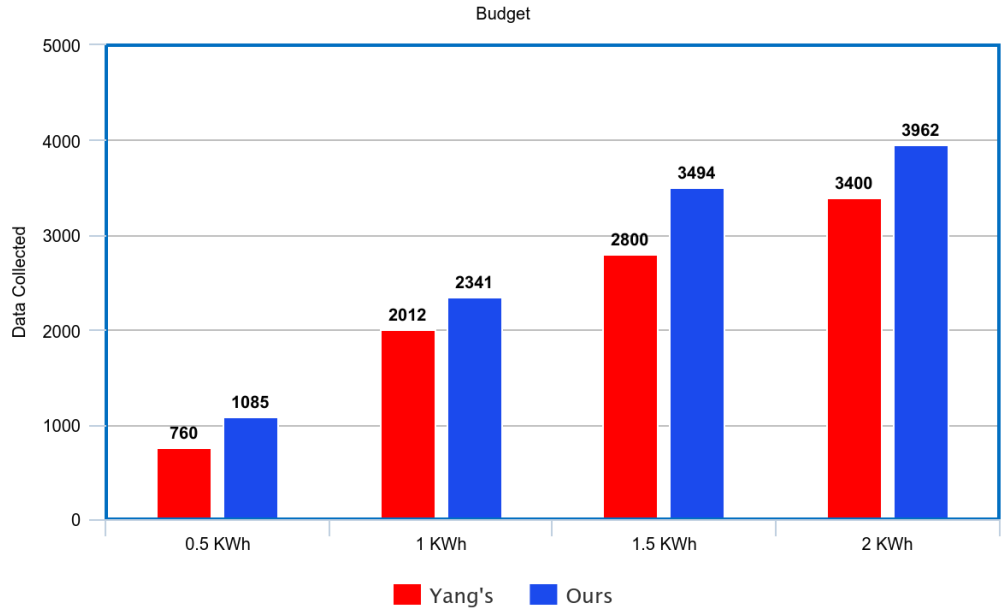
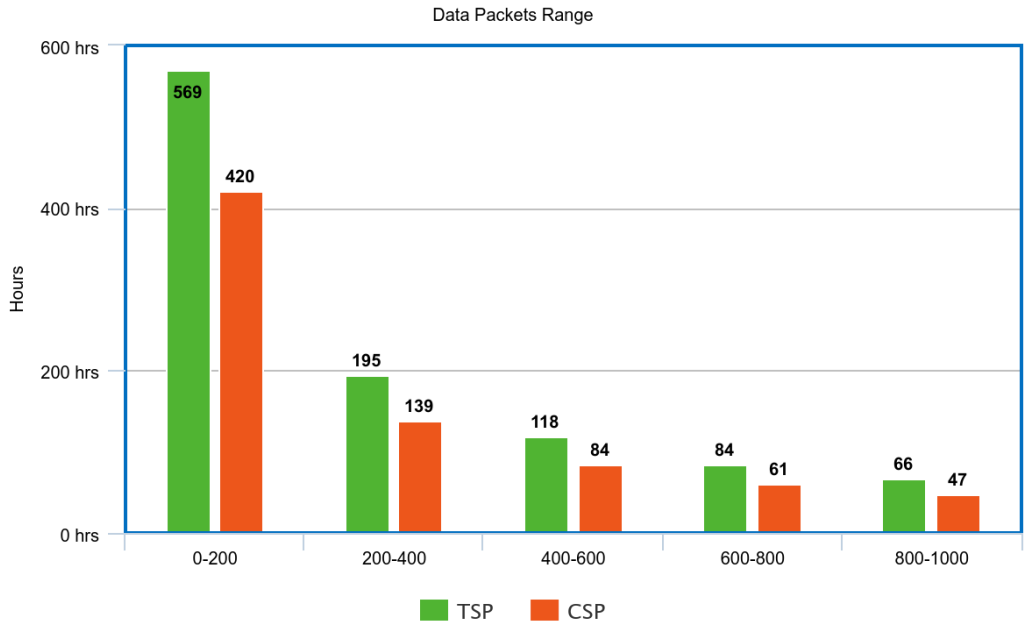


Figure 26

Network Longevity



Network Longevity In Figure 26, it is evident that the network exhibits greater longevity when utilizing the TSP approach for data collection. This is attributed to the fact that in TSP, sensor

nodes only expend energy during data sensing activities. Conversely, in CSP, when the robot reaches a node, neighboring nodes transmit data wirelessly, leading to energy expenditure from each node's battery power. This relationship is mathematically expressed in Chapter 3, under problem formulation. The results are observed across a range of data packet quantities, where higher data packet counts correspond to increased energy consumption for transmission/ reception, consequently resulting in reduced network longevity.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

We introduced an algorithmic problem termed budget-constrained TSP (BC-TSP), which finds application in various robotic scenarios where robots are assigned tasks under battery power constraints. Such contexts include robotic sensor networks, electric cars in ride-sharing services, and automated warehouses. We developed two greedy algorithms and a multi-agent reinforcement learning (MARL) algorithm to address BC-TSP. Our experimental results demonstrate that the MARL algorithm outperforms the handcrafted greedy algorithms and the minimum spanning tree algorithm in terms of prizes collected. Moreover, it closely approaches the optimal Integer Linear Programming (ILP) solution with a negligible difference of just 0.2%. Furthermore, our formulation of BC-CSP (budget-constrained covering salesman problem) indicates an enhanced data collection capability compared to existing greedy algorithms, achieving a 40% increase. The comparison between TSP and CSP approaches highlights TSP's superior efficiency in conserving network energy and prolonging network longevity, hence preserving the data.

In our future research endeavors, we aim to extend our comparisons by including a deep reinforcement learning (DRL)-based approach. DRL, leveraging neural network-based function approximation algorithms in conjunction with reinforcement learning (RL), has emerged as a potent framework for addressing combinatorial optimization challenges (Chen et al., 2016; Heinzelman et al., 2000; Current & Schilling, 1989; Luo & Hubaux, 2005; Ma et al., 2013; Sutton & Barto, 2020). DRL exhibits the capability to handle intricate states and decision-making processes for agents effectively. However, in this current study, we have opted for a multi-agent reinforcement learning (MARL) approach over DRL. This decision stems from our

desire to maintain transparency and control over the learning process, which may be perceived as a "black box" in DRL, especially concerning agent learning within neural networks.

REFERENCES OR WORKS CITED

Basagni, S., Boloni, L., Gjanci, P., Petrioli, C., Phillips, C. A., & Turgut, D. (2014). Maximizing the value of sensed information in underwater wireless sensor networks via an autonomous underwater vehicle. *Proc. of INFOCOM*.

Chen, L., Wang, W., Huang, H., & Lin, S. (2016). On time-constrained data harvesting wireless sensor networks: Approximation algorithm design. *IEEE/ACM Transactions on Networking, 24(5), 3123-3135*

Coutinho, R. W. L., Boukerche, A., & Guercin, S. (2019). Performance evaluation of candidate set selection procedures in underwater sensor networks. *Proc. of IEEE ICC 2019.Networking, 24(5), 3123–3135.*

Current, J. R., & Schilling, D. A. (1989). The covering salesman problem. *Transportation science, 23*, 208–213.

De Francesco, M. D., Das, S. K., & Anastasi, G. (2011). Data collection in wireless sensor networks with mobile elements: A survey. *8(1)*

Garaffa, L. C., Basso, M., Konzen, A. A., & de Freitas, E. P. (2023). Reinforcement learning for mobile robotics exploration: A survey. *IEEE Transactions on Neural Networks and Learning Systems, 34*(8), *3796–3810*.

Ghosh, P., Gasparri, A., Jin, J., & Krishnamachari, B. (2019). Robotic Wireless Sensor Networks. *Springer International Publishing*.

Gu, Y., Ren, F., Ji, Y., & Li, J. (2016). The evolution of sink mobility management in wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials, 18*(1), *507–524*.

Guo, S., Wang, C., & Yang, Y. (2014). Joint mobile data gathering and energy provisioning in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing, 13*(12), *2836–2852*.

Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proc. of HICSS 2000.*

Kim, D., Xue, L., Li, D., Zhu, Y., Wang, W., & Tokuta, A. O. (2017). On theoretical trajectory planning of multiple drones to minimize latency in search-and-reconnaissance operations. *IEEE Transactions on Mobile Computing, 16*(11), *3156–3166*.

Littman, M. L. (2001). Value-function reinforcement learning in Markov games. *Cognitive Systems Research, 2*(1), *55–66*.

Liu, S., & Sun, D. (2014). Minimizing energy consumption of wheeled mobile robots via optimal motion planning. *IEEE/ASME Transactions on Mechatronics, 19*(2), *401–411*.

Liu, X., Qiu, T., Zhou, X., Wang, T., Yang, L., & Chang, V. (2020). Latency-aware path planning for disconnected sensor networks with mobile sinks. *IEEE Transactions on Industrial Informatics, 16(1), 350–361.*

Ma, M., Yang, Y., & Zhao, M. (2013). Tour planning for mobile data-gathering mechanisms in wireless sensor networks. *IEEE Transactions on Vehicular Technology, 62(4), 1472–1483*.

Rawat, P., Singh, K., Chaouchi, H., & Bonnin, J. M. (2014). Wireless sensor networks: A survey on recent developments and potential synergies. *The Journal of Supercomputing, 68, 1–48.*

Salarian, H., Chin, K. W., & Naghdy, F. (2014). An energy-efficient mobile-sink path selection strategy for wireless sensor networks. *IEEE Transactions on Vehicular Technology, 63*(5), *2407–2419*.

Sutton, R. S., & Barto, A. G. (2020). Reinforcement Learning, An Introduction. *The MIT Press.*

Wang, C., Guo, S., & Yang, Y. (2016). An optimization framework for mobile data collection in energy-harvesting wireless sensor networks. *IEEE Transactions on Mobile Computing, 15*(12), *2969–2986*.

APPENDIX A: GITHUB REPOSITORY LINK

https://github.com/skapa-xd/Research