2014

Ant Colony Optimization for Jointly Solving Relay Node Placement and Trajectory Calculation in Hierarchical Wireless Sensor Networks

K. Raiyan Kamal
University of Windsor

Recommended Citation

Kamal, K. Raiyan, "Ant Colony Optimization for Jointly Solving Relay Node Placement and Trajectory Calculation in Hierarchical Wireless Sensor Networks" (2014). *Electronic Theses and Dissertations.* Paper 5054.

Ant Colony Optimization for Jointly Solving Relay Node Placement and Trajectory Calculation in Hierarchical Wireless Sensor Networks

By

K Raiyan Kamal

A Thesis

Submitted to the Faculty of Graduate Studies

through the School of Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

2014

Ant Colony Optimization for Jointly Solving Relay Node Placement and Trajectory Calculation in Hierarchical Wireless Sensor Networks

By

K Raiyan Kamal

APPROVED BY:

01/17/2014

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Given the locations of the Sensor Nodes in a Wireless Sensor Networks (WSN), finding the minimum number of Relays required and their locations such that each sensor is covered by at least one relay is called the Relay Node Placement(RNP) problem. Given the locations of the relays, finding an optimized trajectory for the Mobile Data Collector(MDC) is another important design problem of the WSN domain. Previous researchers have shown that jointly solving different design problems in the WSN domain often leads to better overall results. In recent years, Ant Colony Optimization (ACO) have emerged as an effective tool for solving complex optimization problems. An ACO based approach for solving the joint problem of Relay Node Placement & Trajectory calculation(RNPT) is proposed in this thesis. We also present a deterministic, and a Continuous Ant Colony Optimization ($ACO_{\mathbb{R}}$) approach for refining the trajectory produced by the ACO approach.

DEDICATION

To Humanity.

ACKNOWLEDGMENTS

During the course of the research work presented in this thesis, I received generous help from various individuals.

First of all, I would like to express my deepest gratitude to my supervisors, Dr. Subir Bandyopadhyay, and Dr. Arunita Jaekel. Without their valuable advice, guidance, and patient encouragement, this research work would not have been completed.

I would also like to express gratitude to my committee members, Dr. Yash Aneja, and Dr. Luis Rueda for their precious time to read my thesis and for their valuable comments and suggestions on my work.

Much appreciation also goes to my peers who have aided me with comments, critiques, and discussions. I am also grateful to the staff of the School of Computer Science at the University of Windsor who have helped me out on numerous occassions from filling out forms to installing software modules on the servers.

Last but not least, I would like to thank the free and open source software community, whithout whose generous contributions, none of the software tools, operating system or programming languages used to carry out my research would have existed.

Contents

List of Tables

List of Figures

List of Algorithms

Nomenclature

ACO	Ant Colony Optimization
$ACO_{\mathbb{R}}$	Continuous Ant Colony Optimization
AS	Ant System
BS	Base Station
CnOP	Continuous Optimization Problem
COP	Combinatorial Optimization Problem
EAS	Elitist Ant System
MDC	Mobile Data Collector
MMAS	Max Min Ant System
RNP	Relay Node Placement
RNPT	The joint problem of Relay Node Placement and Trajectory calculation
TSP	Traveling Salesman Problem
TSPN	Traveling Salesman Problem with Neighborhood

WSN	Wireless Sensor Network
α	algorithmic parameter for importance of pheromone trace
β	algorithmic parameter for importance of heuristic information
$\mathcal{R}$	set of potential relay locations
ρ	Pheromone evaporation rate
τ	Pheromone matrix
C_s	Cost of a solution s
L	size of the candidate list
l_s	Number of relays used in the solution s
m	number of ants employed
N	number potential relay node locations
n	number of sensors
$N(i)$	neighborhood of i
s	A solution instance

1. Introduction

Wireless Sensor Networks (WSN) are ad-hoc networks consisting of numerous, low-powered and multi-functional sensing devices called *Sensor Nodes,* which are capable of sensing one or more physical parameters of the ambient environment and maintaining communication with other members of the network through the wireless medium. Besides the sensors, a WSN may include some specialized entities such as the Base Station(BS). The BS acts as a point of authority for the entire network where the data generated from different sensors are accumulated, processed and made accessible to the users. A WSN is usually deployed to monitor an area of interest known as the sensing field. The dimension of the sensing field can range from the size of a small house to several square kilometers[68].

The origins of WSN can be traced back to the 1950s when the United States Military launched a project, which later came to be known as Sound Surveillance System(SOSUS)[27], for detection and tracking of submarines during the cold war period. The SOSUS consisted of submerged acoustic sensors – hydrophones – communicating with reception stations located in coastal areas. This sensing technology is still in service, but the mission has shifted to the peaceful and constructive purpose of monitoring undersea wildlife and volcanic activity.

Advances in semiconductor, networking and material science technologies in recent decades have made large-scale WSNs a possibility. Together, these technologies have combined to enable a new generation of WSNs that differ greatly from wireless networks developed and deployed a decade or more ago. Today's state-of-the-art WSNs have lower deployment and maintenance costs, last longer, and are more rugged. There has been significant advancement in accessibility as well. Off-the-shelf components are available for today's WSN designer that are capable of bridging with mainstream networks such as the Internet, to upload the data to a cloud service[2] or provide means of controlling various operational parameters of the WSN from a remote location. The next big step to formally explore the challenges in implementing a distributed network of wireless sensors was taken by the United States Defense Advanced Research Projects Agency(DARPA) by launching the Distributed Sensor Network (DSN) program in the 1980s. The DSN project involved several universities besides the US Military, resulting in an increase of civilian scientific and engineering research.

The early WSNs consisted of bulky sensing and communication machinery. Applications of WSN was limited to military, and heavy industrial settings. With the advancements in semiconductor technology, networking and material science in the 1990s, this started changing rapidly. As the technology matured, the range of application of WSN grew to include smart home health care[5], industrial control and monitoring[89, 35], precision agriculture[82], wildlife tracking and habitat monitoring[57], disaster relief management[15] etc.

1.1. Motivation

Modern WSN are deployed in a wide range of scenarios. The sensing field can be somewhere as close as an urban household, in the case of monitoring power consumption in a smart home or as remote as a volcanic site miles away from research base. The environmental phenomena being monitored can be either man-made such as monitoring electricity and/or voltage levels in a smart grid, pressure of fluids in an industrial setting, presence of pollutants in the air, structural integrity of a building or it can be purely natural such as the behavior of wild animals, rainfall, seismic or volcanic activity.

The advantages of using WSN in such a setting: convenience in data collection and cost reduction. For any of the examples mentioned above, a conventional sensing system could very well achieve the goal of sensing environmental parameters using same or similar set of components connected by a wired network. However, the establishment cost and maintenance cost of such a wired network, which is not trivial by any means, can be eliminated altogether if a WSN is used. Since this cost grows proportionally with the distance between the sensing field and the BS, deploying a WSN to monitor a remote sensing field can result in significant cost saving.

In the case of remote sensing applications, random deployment of sensors is the only feasible option. In this kind of deployment maintaining connectivity of the network cannot be guaranteed. A special kind of component called the *Relay Node[10, 11]* is introduced to face this challenge. The relay node is essentially a sensor node provisioned with higher capacity energy source, memory and processor. The relay nodes are tasked with collecting and forwarding the data from the sensor nodes.

However, since the power dissipation in radio communication grows with the distance between the sender and the receiver, further energy saving can be achieved by provisioning the BS with the capacity of mobility. Such a BS is called a Mobile Data Collector(MDC)[72]. In such a WSN, the data from sensor nodes are buffered at relay nodes and the MDC which is capable of navigation across the sensing field, visits the relay nodes to download the buffered data. The MDC is assumed to know the locations of the relay nodes, from which a trajectory is calculated to sequentially visit the relay nodes. Under this scheme, the maximum distance over which a relay node transmits is significantly reduced, resulting in energy saving at the relay nodes as well as at the sensor nodes.

Since the relay nodes are provisioned with higher capacity hardware, they are more expensive compared to sensors nodes. For this reason, it is an important design problem in the domain of WSN to find out the minimum number of relays and their locations such that each sensor node in the network is connected to at least one relay node. This problem is known as the Relay Node Placement Problem(RNP). Another important design consideration is reducing the trajectory of the MDC. A shorter trajectory implies smaller interval between two successive visit to a relay node. This improves the timeliness of the data, and also reduces the required buffer size.

1.2. Solution Outline

From the work of previous researchers, it was found that jointly solving the design problems in the WSN domain often lead to improved result than separately solving the design problems[55, 56, 8, 12]. However, both the RNP and trajectory calculation problems have been shown to belong to the class of problems called NP-hard[76, 28, 30, 69]. Heuristic algorithm often deliver acceptable solutions within reasonable time frame in such cases. A population based meta heuristic called Ant Colony Optimization(ACO) has emerged in recent years as a powerful tool for tackling complex optimization problem [73]. In this thesis, an ACO approach is proposed for jointly solving the RNP and trajectory calculation problem for the MDC in a WSN.

1.3. Organization

The remainder of this thesis is organized as follows. In Chapter 2, an introduction to Wireless Sensor Networks, and Ant Colony Optimization as a tool for solving design problems in WSN is presented. In Chapter 3, an approach for jointly solving relay node placement and trajectory calculation of MDC in a hierarchical WSN is proposed. Experimental results and findings are presented in Chapter 4. The thesis is concluded with a summary and direction for future works in Chapter 5.

2. Background Review

A detailed discussion on Wireless Senor Networks and some important design problems are presented in this chapter. Further, the *Ant Colony Optimization(ACO)* meta-heuristic is introduced as a tool for jointly solving the relay node placement and trajectory calculation for a Mobile Data Collector in a 3-tier Wireless Sensor Network.

2.1. Wireless Sensor Networks

Wireless Sensor Networks (WSN) are ad-hoc networks consisting of a number of small sensing devices, called sensor nodes, and a Base Station (BS). The sensor nodes (or sensors, for short) are capable of sensing one or more physical parameters of the environment and communicating with each other and the BS through wireless medium[3, 16]. The members of a WSN work collaboratively to sense one or more physical parameters of the environment and forward the collected data to the BS, which serves as as a central data repository.

A WSN is deployed to monitor a geographical area, called the sensing field. The sensing field can be a remote location[85], or in a hazardous (due to pollution, or presence of radioactivity[52]) environment, or discourages human intervention e.g., wildlife habitat[57, 77]. Due to such restrictions, replacing the faulty sensors is not feasible. Therefore, the sensors are designed as low-cost, disposable units of small dimension. Small dimension poses strict constraints on processor, memory and power source. Due to such operating conditions, efficient power management is of critical importance to the lifetime of a sensor and the WSN as a whole.

The BS on the other hand is not power constrained. It can be stationary or mounted on a vehicle capable of navigating through the sensing field. Such a non-stationary BS is called a Mobile Data Collector (MDC). Besides being the central repository for the data collected by a WSN, the BS also serves as a data processing centre, and as an access point for the information through a conventional network such as the Internet. A schematic of the data flow from a WSN to the end user can be seen in Figure 2.1.

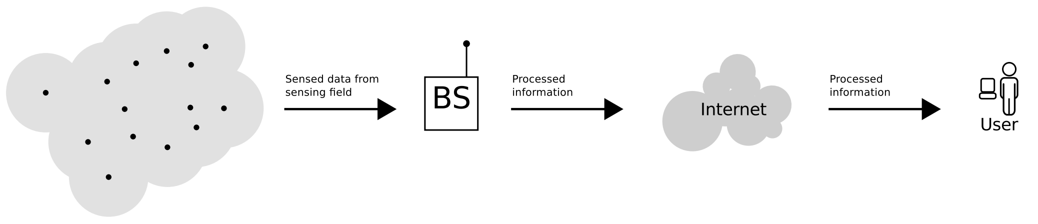
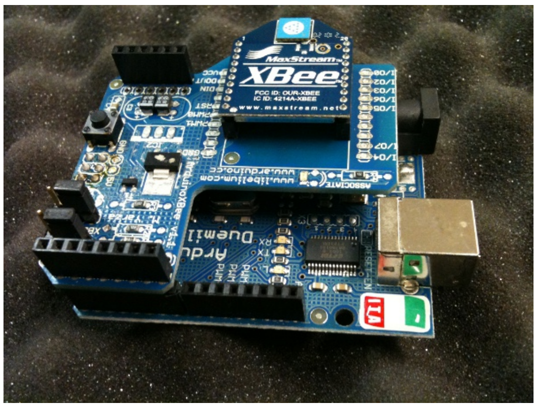
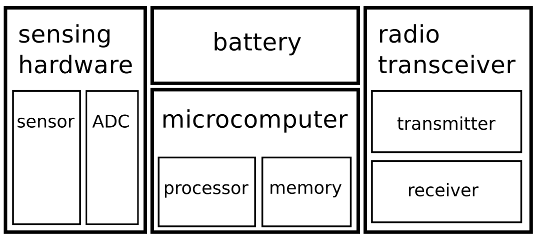


Figure 2.1.: Data flow from a WSN to an end user.

2.1.1. Sensor Nodes

The sensor nodes are the building blocks of a WSN. A typical sensor node and its schematic drawing are shown in figure 2.2a and 2.2b. A sensor node consists of a microcomputer, sensing hardware, radio transceiver and a battery[3]. The sensing hardware consists of an Analog to Digital Converter(ADC) and one or more sensors for measuring physical parameters such as light, temperature, humidity, vibration etc of the ambient environment. The ADC converts the analog signals from the sensor into digital values which are then passed on to the microcomputer. The microcomputer consists of a microprocessor, and a memory unit. The data collected by sensors are buffered in the memory until they are forwarded to the BS. Buffered data can be transmitted directly to the BS or routed through other members of the network acting as intermediaries. The transceiver maintains communication with the network; it consists of radio transmitter, and receiver circuitry. All of these components are connected to the battery from which they draw the necessary energy to operate. The radio transceiver, sensors, batteries and other units are available as off-the-shelf components[2].


(a) A state-of-the-art sensor node.



(b) Schematic drawing of a sensor node.

Figure 2.2.: Sensor node.

The sensor nodes are designed as autonomous units. Recharging or exchanging the batteries of individual sensor nodes is generally considered too costly to carry out. Once the limited energy of the battery is completely dissipated, a sensor node will be out of operation and lose its functionality[80]. Therefore, sensors are designed as disposable, low-cost units. The requirement of small dimension puts constraints on the size (and capacity) of the battery on board. Since the communication range of a wireless device is directly related to available energy, the sensor nodes typically have a limited communication range as well.

2.1.2. Sensor Node Deployment

Sensor nodes are normally deployed inside or very close to the phenomenon of interest in order to ensure effective sensing. Sensors within a sensing field can be placed either in a pre-determined fashion or in a random distribution. The pre-determined placement applies to situations where the sensing field is accessible. This strategy achieves better coverage[86, 40], but relies on prior knowledge of the sensing field. However, in many real life cases, e.g. in hostile environment such as battle field or polluted area, randomly deploying sensor nodes is more practical and sometimes the only possibility[3, 68]. Random deployment is also faster compared to pre-determined placement. However, it requires self-organized routing schemes and distributed network algorithms to be incorporated in to sensor networks, which are relatively complex. Despites these challenges, the random deployment is more popular in real life application due to the practical aspect[79, 41, 43].

2.1.3. Energy Model of WSN

The first order radio model provides a metric for the energy dissipation at each node of a WSN[38]. According to this model, energy dissipation is calculated for communicating one bit of information. Amount of energy dissipated in transmitting one bit, E_{tx} can be calculated by:

$$E_{tx} = E_t + E_d \times d^n \tag{2.1}$$

Here, E_t is the energy dissipated in transmission circuitry, E_d is the energy dissipated in transmission, d is the distance between transmitter and receiver, and n is the path loss component which is a physical property of the medium. The value of n is 2 for air.

The amount of energy dissipated in receiving one bit of information is calculated by:

$$E_{rx} = E_r \tag{2.2}$$

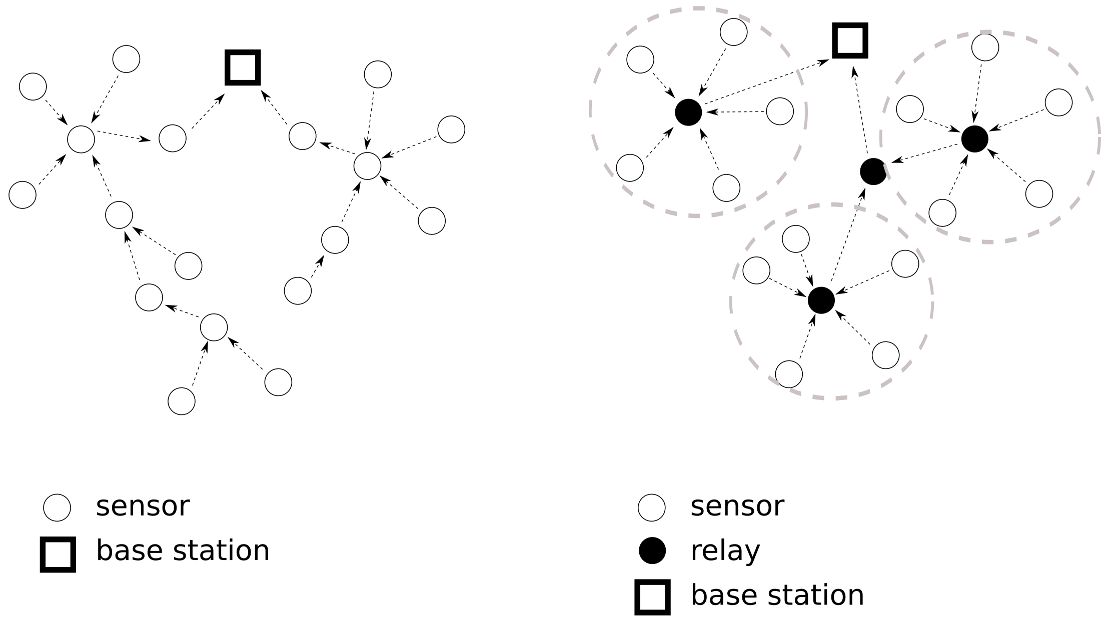
Here, E_r is the energy dissipated in reception circuitry.

Although in theory it is possible to transmit from a sensor over a distance as large as the battery permits, in practice a sensor is restricted to transmit within a pre-specified distance called the communication range, denoted by r. A two-dimensional disk of radius r centered at a sensor defines the *region of influence* of that sensor. A sensor can transmit to another sensor (or the BS) located inside its region of influence.

2.1.4. Network Model of WSN

Based on their network architecture, WSNs can be classified as either flat or hierarchical networks. In flat sensor networks, all sensors nodes are made identical and are assigned the same roles. Besides sensing the environment, the sensors in a flat network are tasked with forwarding their sensed data and routing data from other sensors towards the BS. A typical flat WSN is shown in Figure 2.3a.

In a hierarchical WSN, the members of the network are organized in different tiers; each tier performing some specific tasks. The lowest tier or first tier consists of sensors nodes, responsible for sensing the environment. A second tier consists of nodes tasked with the routing and forwarding of data sensed by the sensors at the first tier. These nodes are called *relay nodes* (or *relays*, for short). The sensors are grouped in clusters, and each cluster is headed by a relay[10, 11]. Each sensor usually belongs to only one cluster and communicates directly to its cluster head. All the



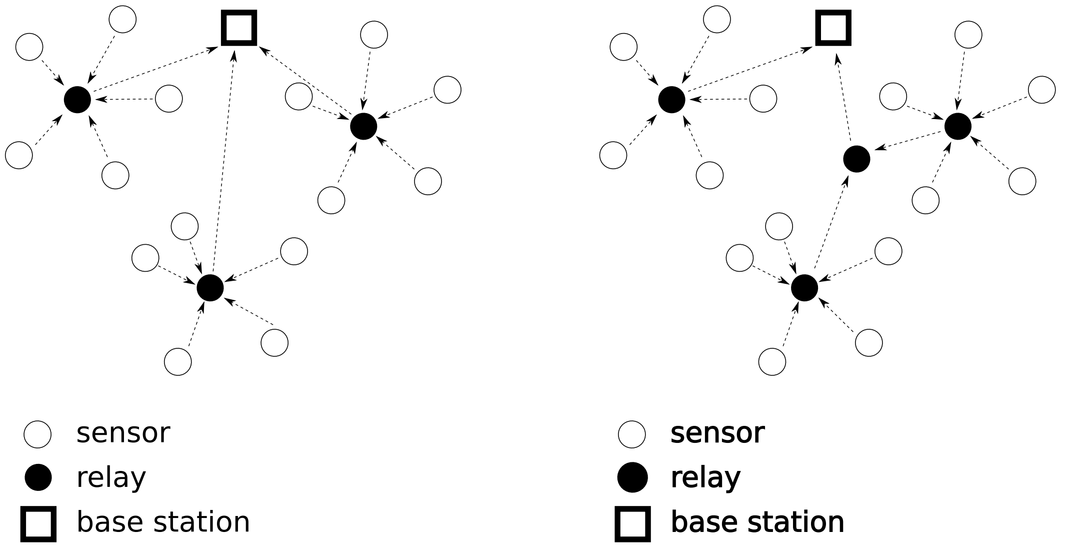
(a) Flat WSN. **(b)** Hierarchical WSN.

Figure 2.3.: Flat, and hierarchical WSN

data from the sensors in a cluster are thus collected and buffered in the respective cluster head. The cluster head forwards the buffered data using an appropriate routing scheme towards the BS[36]. An example of a 2-tiered WSN is shown in Figure 2.3b.

Compared to flat architecture, hierarchical model achieves prolonged network lifetime[78]. In a 2-tiered WSN, for example, sensor nodes in the lower tier are relieved from the burden of routing and forwarding; this reduces the energy consumption of these nodes[78, 36, 10]. Due to such advantage, hierarchical architecture has gained increased popularity in the research and development of sensor networks.

Since the relays are required to hold more data and transmit over a larger distance, compared to sensors, they are usually equipped with higher capacity memory and battery. This also makes them more expensive than sensors. For this reason the relays are not designed as disposable units, unlike the sensors. This makes it desirable to keep the number of relays used in a hierarchical WSN as small as possible, to lower the setup and maintenance cost. This gives rise to a well known design problem in the WSN domain, called the *relay node placement problem.* This problem is discussed in detail in section 2.2.



(a) Single-hop routing. **(b)** Multi-hop routing.

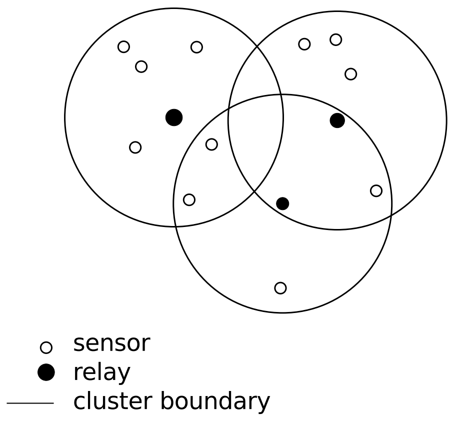
Figure 2.4.: Routing schemes in a 2-tier WSN

The relays in the upper tier of a 2-tier network can follow either a *single-hop* or *multi-hop* routing scheme to forward the data collected from the lower tier. In a single-hop scheme, the relays communicate directly with the BS, as shown in Figure 2.4a. In this scheme, the relays located far away from the BS dissipate more power than the ones located nearby, as can be explained by the first order radio model. In multi-hop routing scheme, the relays located near the BS act as intermediary between the BS and other relays located far away from the BS. Figure 2.4b shows a 2-tier WSN using multi-hop routing. In this scheme, the intermediary relays have to transmit large volume of data towards the BS. As a result, energy dissipation is higher in the relays nearer to the BS than the ones that are farther. Depending on the type of routing scheme used, either the relays close to the BS or the ones far from the BS are depleted of all energy sooner than the rest, reducing the operating lifetime of the network as a whole.

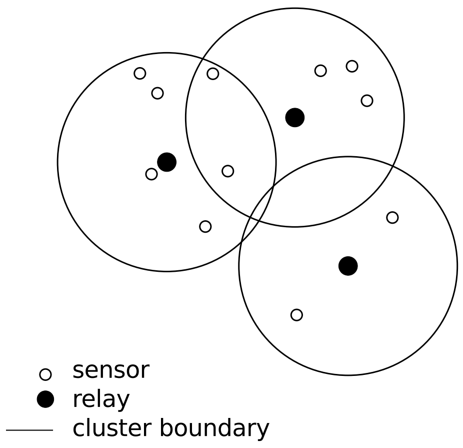
2.2. Relay Node Placement Problem

Relays are more expensive than sensors and unlike sensors they are not designed as disposable units. Thus, the number of relays used in a network directly contributes to the establishment and maintenance cost of the network. Therefore, reducing the number of relays used while still maintaining full coverage of the network in a WSN is an important design consideration. This concern is addressed in the *Relay Node Placement(RNP)* problem. Given the locations of sensor nodes in the sensing field, the RNP problem asks to find the minimum number of relays, and the locations to place the relays such that each sensor should be connected to at least one relay. Multiple solutions can exist for a particular distribution of sensors nodes. The non-uniqueness of the solutions is illustrated in Figure 2.5 with two different relay placements for the same scenario. Both solutions use same number relays (3) but their locations are different.

The RNP problem has been shown to be NP-hard[76, 28]. According to the computational complexity theory, a problem is classified as NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem. An NP-problem is one which is solvable in polynomial time by a nondeterministic Turing



(a) Placement a.



(b) Placement b.

Figure 2.5.: Different solutions of the placement problem for the same scenario.

machine[18, 83, 84]. An NP-hard problem quickly becomes intractable as the input size (number of sensors in this case) increases. As a result, simple exhaustive search is not a feasible solution approach for this problem. Previous researchers have proposed approximation or heuristic algorithms to find near optimal solutions of the RNP within a reasonable time limit[39, 54, 78, 61].

A joint solution for RNP with *Energy Provisioning* in a 2 tier WSN is presented in[39]. The Energy Provisioning problem refers to finding a given number of senors which can be provisioned with a given amount of extra energy so that the overall network lifetime would be increased. The joint problem has been modeled as a *Linear Programming(LP)* problem. A heuristic called *Smart Pairing and INtelligent Disc Search (SPINDS)* has also been proposed. A set of approximation algorithms for RNP is presented in[54]. An 7-approximation algorithm (with upper bound on results being 7 times that of the optimum value) to solve the RNP has been presented. This algorithm also ensures that there is a path consisting of sensors or relays between every pair of sensors in the sensing field. This problem is referred to as the *Single-Tiered Relay Node Placement Problem.* Another version of the problem is the Two-Tiered Relay Node Placement Problem, where the path between every pair of sensor nodes consists solely of relays, has been solved with (5+ϵ)-approximation algorithm. A polynomial time approximation algorithm with constant bounds for placing relay nodes in large scale two-tiered networks is presented in [78]. The proposed solution ensures that each sensor node is covered by at least two relay nodes and there exist two node-disjoint paths between each pair of relay nodes in the network, this problem is referred to as *2-Connected Relay Node Double Cover (2CRNDC)* problem. The authors of [78] also presented a similar solution for *Connected Relay Node Single Cover (CRNSC)* problem where, the condition is that each sensor should be connected to a relay and the sub network consisting of relays should also be connected. A relay node placement strategy for WSN with bi-connectivity requirement and under locations constraints for placing the relays is presented in [61]. This work also includes a framework for $\mathcal{O}(1)$ approximation algorithm for solving the RNP problem.

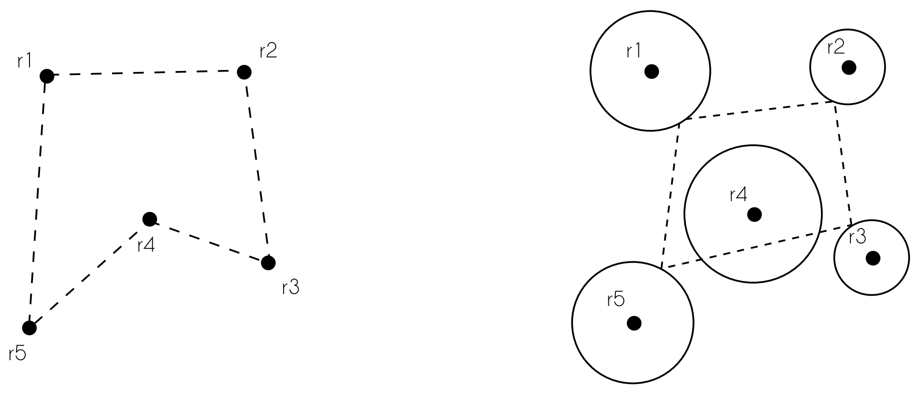
2.3. 3-Tier WSN and Trajectory Calculation for MDC

The limitations of a 2-tier WSN with stationary BS can be overcome by replacing the BS with a *Mobile Data Collector (MDC).* The MDC is basically a BS mounted on a vehicle capable of navigating across the sensing field[72]. This component alone forms the third tier of a hierarchical WSN. In this scheme, the data collected from sensors are buffered in the the relays acting as cluster heads. In such a 3-tier WSN, the MDC moves at a constant speed along a pre-calculated trajectory to visits each relay in sequence to download the buffered data. Once the buffered data from a relay is downloaded to the MDC, its buffer is cleared and becomes ready to hold new incoming data from the sensors in its respective cluster.

It has been shown that inclusion of an MDC can improve the performance of a WSN in terms of network lifetime, coverage, connectivity and fault-tolerance[9, 29, 45, 55, 62, 72, 44, 34]. Since the MDC can move within the sensing field, the need for dense deployment of sensors and relays to ensure coverage and connectivity is eliminated. The involvement of MDC in the network also improves the lifetime of the network because the nodes would transmit to much shorter distance, leading to less power dissipation at individual nodes.

Introducing an MDC into the network brings additional challenge of calculating a trajectory for the MDC. Calculating the trajectory of an MDC is of significant importance because shorter trajectory of MDC implies shorter interval between two successive visits to a relay. This improves timeliness of the collected data, and also leads to reduced buffer size in the relays[12]. Given the locations of a set of relays in the sensing field (a solution of the placement problem), calculating the trajectory of the MDC in a 3-tier WSN requires finding the shortest trajectory that allows the MDC, starting from a point in the sensing field, to visit each relay at least once before coming back to the starting point. A relay is considered visited by the MDC when the data buffered in relay has been downloaded to the MDC. This problem is closely related to the *Traveling Salesman Problem(TSP)*, a well studied problem in Computer Science[18].

The Traveling Salesman Problem problem asks, given a set of entities called called cities, and pairwise distances between the cities, what is the shortest possible tour that visits each city exactly once and returns to the starting city[18]. Figure 2.6a shows a TSP instance with 5 cities. This problem belongs to the computational class NP-hard[18]. Practical application of TSP and its variations can be found in diverse range of domains including vehicle routing [7], computer wiring [49], job sequencing [31], crystallography [13] etc. Due to its widespread applicability, TSP is one of the most intensively studied combinatorial optimization problem in Computer Science[48]. Since the TSP quickly becomes intractable as the number of cities grow, approximation and heuristic algorithms[58] have gained popularity for solving large TSP instances.



(a) An instance of TSP. **(b)** A TSPN instance extended from TSP.

Figure 2.6.: Comparison of TSP and TSPN tours.

A generalized version of the TSP, the Traveling Salesman Problem with Neighborhoods (TSPN), extends the TSP to the case where cities are defined as regions on the plane. A city is considered visited in TSPN as long as at least one point from its corresponding region is reached[70]. The TSP instance in Figure 2.6a can be extended to a TSPN instance by considering the cities as circles of different radii centered at the given points. An example TSPN instance is shown in Figure 2.6b. The dashed line represents the TSPN tour. A TSPN tour has a shorter tour length than the corresponding TSP tour, as illustrated in Figure 2.6. Applications of TSPN include VLSI routing[66], communication networks[42] etc.

In the context of 3-tier WSN, The TSPN problem is of higher importance compared to the TSP, because the optimum trajectory of an MDC is essentially a TSPN tour. A feasible trajectory of the MDC is a solution instance of the TSP problem formed by considering the relays as cities and their pairwise distances defined by the Euclidean distance. Following such a trajectory, the MDC stops at each relay to download its buffered data. However, due to the wireless communication capability of the relays and the MDC, its is not necessary for the MDC to be on the exact same spot as a relay in order to download data from it; the MDC is only required to be sufficiently close (to the relay) to initiate communication. This can be ensured as long as the MDC is at the boundary or inside the relay's region of influence. Therefore, the trajectory calculation can be solved as a TSPN problem where each relay is a city with the neighborhood defined as a disk of radius equal to the relay's range of communication and centered at the relay. Following such a trajectory, the MDC is likely to traverse much shorter length as illustrated in the side by side comparison in Figure 2.6.

Compared to TSP, which is a combinatorial optimization problem, the TSPN is an optimization problem with two distinct components: a combinatorial and a continuous one. The combinatorial aspect of TSPN is to decide the order in which the neighborhoods need visiting. The continuous aspect is to decide for each neighborhood which point within the neighborhood needs visiting. Like TSP, TSPN is also classified as NP-hard[30, 69]; the running time of a deterministic solution can be prohibitively large. The TSPN problem was first studied by Arkin and Hassin[4], they proposed an $\mathcal{O}(1)$ approximation algorithm for solving TSPN where neighborhoods are: parallel unit-length segments, or translates of a convex polygons or, (more generally) shapes with diameter segments that are parallel to a common direction and a ratio between the longest and the shortest diameter that is bounded by a constant. An approximation algorithm is presented in[59] for solving TSPN in $\mathcal{O}(\lg k)$ time where neighborhoods are arbitrary polygons. A polynomial-time constant-factor approximation algorithm for disjoint convex fat neighborhoods of arbitrary size is presented in [20].

Several researchers have investigated the trajectory calculation for one or more MDCs in WSN. The range of approaches include *Linear programming(LP)*, deterministic algorithms, heuristic, and approximation algorithms. A 3-tier architecture for sparse WSN containing multiple mobile entities called *Data Mules* has been proposed in[72]. The data mules, playing a similar role as MDC, are assumed to walk randomly within the sensing area and no trajectory calculation is involved. A partitioning based algorithm for a network containing multiple MDCs is presented in[34]. This approach suggests that the MDCs should visit each individual sensor node in order to collect the buffered data therein. An LP formulation for the joint problems of determining the trajectory of an MDC and its sojourn time at different points in the network that leads to the maximum network lifetime is presented in [81]. The MDC in this case moves in direction either parallel to X or Y axis. A more fine grained solution in terms of time and (physical) space is proposed in [47]; the time for the MDC to travel from one point to another along it trajectory is considered in this approach. The *limited hop strategy(LHS)* where only the sensors within a given number of hops away from the MDC transmit their data while the others buffer their data until the sink is accessible to them, is introduced in this work. Heuristics for constructing a trajectory, and for refining the trajectory are also presented. A deterministic solution for fining the trajectory as a TSPN solution from a given TSP is presented in [65]. The quality of the final result in this work is dependent on the initial TSP. A clustering based-genetic algorithm for trajectory calculation of an MDC is presented in [53].

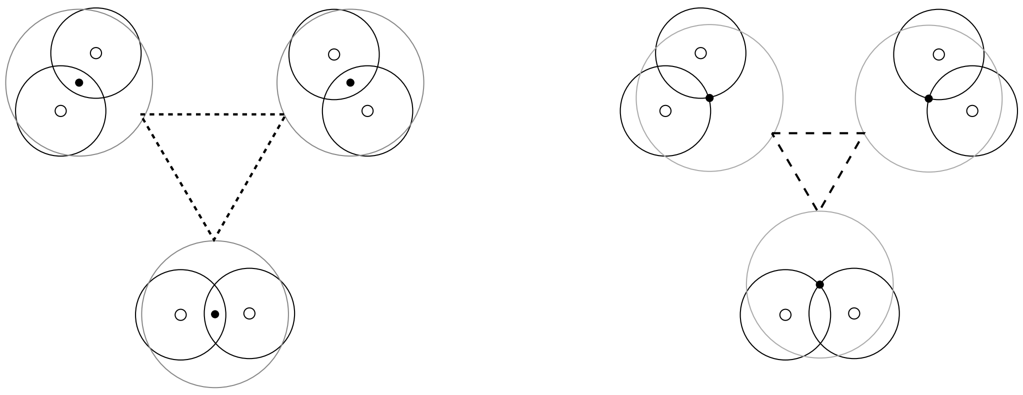
A two phase heuristic for finding the trajectory of an MDC in a sparse WSN is presented in [88]. The WSN is considered sparse in the sense that, for any pair of sensors their regions of influence do not overlap. The trajectory of the MDC, which is also the TSPN solution of the given scenario, is expressed as a set of points called *hitting-points*, a concept introduced in this work; one hitting-point from each relay, and the order of visiting those points. The hitting-point of a relay refers to a point on the boundary of that relay's region of influence, where the trajectory of MDC intersects with the circle defining that relay's region of influence. Since the MDC can start downloading data from a relay as soon as it enters the relay's region of influence, after reaching a particular relay the remainder of the tour is concerned about the next relay to visit and the direction of the trajectory is altered according to do the same. Following this rationale, it is convenient to specify the first intersection point as the hitting-point. The first phase of the approach presented in [88] consists of finding a set of hitting points and their order of visit. The authors of [88] suggested that dedicated TSP algorithms are sufficiently effective for solving large TSP instances and are preferred over evolutionary approaches. In the second phase, an evolutionary approach called the *(1+1) Evolutionary Strategy[71]* is used to refine the choice of hitting points found as the output of the first phase. In this phase, the trajectory is refined by fine tuning the positions of the hitting-points while keeping the order of visiting unaltered.

An *Ant Colony Optimization(ACO)* approach (described in section 2.5, section 2.6, and section 2.7) for solving the trajectory calculation for an MDC in sparse WSN is presented in [17]. The notations and problem formulation in this work is adapted from [88]. Major difference from the approach presented in [88] is that the order of visiting the neighborhoods is not considered fixed in this approach. Although the use of ACO in trajectory calculation presented in [17] is a novel approach, no significant improvement over the work presented in [88] was observed.

2.4. Jointly Solving WSN Design Problems

While designing a 3-tier WSN, the following design goals are of prime importance:

- Every sensor should be covered by at least one relay,
- The number of relays should be optimized,
- The total distance traveled by the MDC to collect data from the relays should be minimized



(a) Result of Separately solving. **(b)** Result of jointly solving.

Figure 2.7.: Comparison of separately solving vs and jointly solving.

The joint *Relay Node Placement and Trajectory calculation (RNPT)* problem encompass all three of these design goals. The RNP and trajectory calculation problems can be solved either separately or jointly. In the former case, the RNP is solved first, then a trajectory for the MDC is calculated, based on the result of RNP. In the latter case, the placement and trajectory calculation is solved as a joint optimization problem. The advantage of jointly solving the two problems is illustrated in Figure 2.7. The solution on left is the result of separately solving the two design problems. The solution on the right is the result of attempting to solve them jointly.

In both cases, number of relays stay the same, but locations vary. It is the different locations of the relays which allows the MDC to traverse a shorter trajectory. Ideally, the non-uniqueness of the placement can be thus leveraged towards finding a shorter trajectory for the MDC.

It has been shown by several researchers in recent years that jointly solving design problems such as RNP, data routing, trajectory calculation etc. in the WSN domain lead to better results [55, 56, 8, 12]. Data routing and trajectory planning has been jointly addressed for improving the network lifetime in[55]. In this work, the sensors are assumed to be deployed within a circular sensing field and the MDC travels along the boundary of that area. An approach to jointly solving the routing and trajectory planning problem in case of constrained mobility i.e., only parts of the sensing field is accessible to the MDC, is presented in [56]. Jointly solving the data routing and RNP problem in a 2-tier hierarchical WSN is shown in [12]. An ILP formulation for jointly optimize the placement and routing is presented in this work. Two heuristics for determining potential relay node locations are also presented in this work.

Both the placement and trajectory calculation problems quickly becomes intractable as the number of sensors increase. In real world setting, a deterministic approach is likely to have prohibitively large running time due to large number of sensors present. Heuristic algorithms on the other hand, can guaranty an acceptable solution within a reasonable time frame. In this thesis, an ACO based approach is proposed for jointly solving the RNP and trajectory calculation for an MDC in a 3-tier WSN. The ACO meta heuristic has shown competitive performance in solving NP-hard problems and has emerged as a powerful tool for solving this class of problem[26].

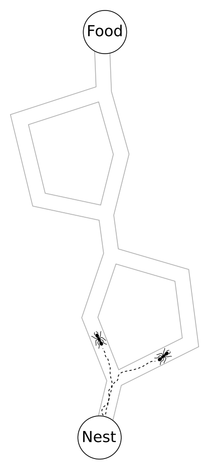
This is the first attempt towards solving the RNPT problem using ACO, to the best of the author's knowledge. The closest previous work can be found in [8], where an

ILP approach for jointly solving the RNP and calculating a load balancing trajectory, and a heuristic for further refining the the solution of the ILP is presented.

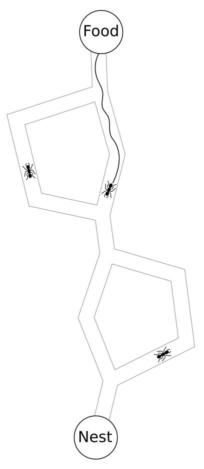
Besides the said ACO approach for jointly solving RNP and trajectory calculation problem, a deterministic heuristic and a continuous ACO approach has been presented for refining a trajectory. Using continuous ACO for calculating the trajectory of and MDC in a non-sparse WSN is also another first attempt to the best of the author's knowledge.

2.5. Ant Colony Optimization

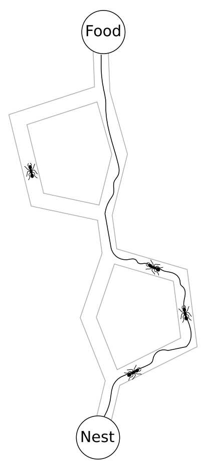
Ant Colony Optimization is a population based meta heuristic derived from observations of the foraging behavior of ants in nature. Although very simple organisms as individuals, capable of only a limited range of actions, a colony of ants working together manages to solve complex problem such as finding an optimized path between their nest and a food source[32]. A special ability called *Stigmergy*, common to ants and other social insects, is essential in this kind of collaborative problem solving by simple agents. Stigmergy refers to the phenomena of indirect communication between the agents of a group, e.g., the ants belonging to a colony, by marking their environment with chemicals called pheromones[14]. A Pheromone is a chemical secreted by an individual that produces a change in the behavior of another individual of the same species; a volatile hormone that acts as a behavior-altering agent[60].



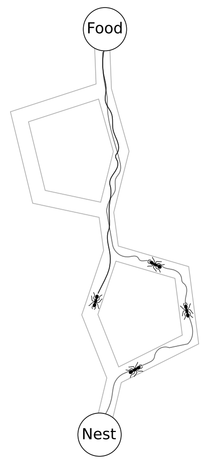
(a) Ants come out of the nest, exploring for food.



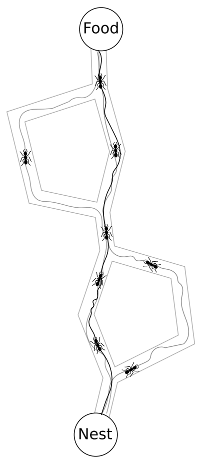
(b) An ant returning to the nest after finding food lays pheromone.



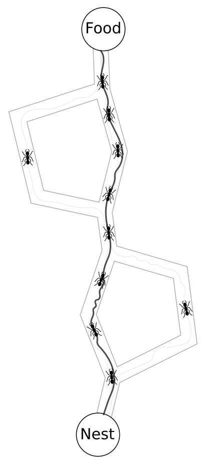
(c) Some of the other ants follow the pheromone trail.



(d) Ants that found food following other paths also lay pheromone.



(e) Pheromone evaporates from long paths over time, making shorter paths more likely to be followed.



(f) High amount of pheromone accumulates on the best path.

Figure 2.8.: Stages of finding path by ants using stigmergy.

The foraging behavior of ants is illustrated in Figure 2.8. To find food, several ants start exploring from the nest (Figure 2.8(a)). This initial exploration is preformed randomly, without any knowledge of the landscape. An ant that finds food, comes back to the nest. On the way back, the ant lays a trail of pheromone on the ground to mark the way to the food source(Figure 2.8(b)). Other ants coming across this pheromone trail takes a biased random decision, based on the amount of pheromone present on the trail, whether to follow the trail. The higher the amount of pheromone, the more likely is the ant to follow the trail. Out of many ants who are exploring, some of them will follow this trail(Figure 2.8(c)). Let the ones deciding to follow the trail be called followers of this trail. Other ants may find food following different paths, these ants lay pheromone as well (Figure 2.8(d)). These trails will have followers too. The followers of a trail returning to the nest after finding food also deposit pheromone on the trail. This causes the amount of pheromone to build up on a promising path, increasing the likelihood of it being followed. However, the pheromones being subjected to evaporation, the intensity of pheromone on a particular trail decreases with time. For paths with shorter length, this does not pose a big problem because by the time an ant on a shorter path reaches the nest after finding food, a good portion of the pheromone laid by this ant is likely to remain. But in the case of a longer path, since it takes more time for an ant following that path to reach the nest, there will be little amount of pheromone remaining at the other end of the path. In fact, depending on the length of the path, all the pheromone laid by the ant may be evaporated. This makes the longer paths less likely to be followed (Figure 2.8(e)). Due to this phenomena, significant amount of pheromone accumulate on the best path(s). In the long run, the best paths direct most of the ants towards food (Figure 2.8(f)). Thus, the random exploration gradually shifts towards guided exploration and eventually toward fully guided expeditions.

2.6. The ACO Meta Heuristic

Inspired by the foraging behavior of ants, the Ant Colony Optimization (ACO) meta heuristic is developed. The ACO was introduced as tool for solving large combinatorial optimization problems in the early nineties [21]. The earliest works on ACO focused on solving the TSP problem [21, 24, 23, 25, 22]. Eventually, ACO found its way in to solving different NP-hard optimization problems such as the graph coloring problem [19], the bin packing problem [51], the quadratic assignment problem [87], the set covering problem [50], the car sequencing problem [33], probabilistic traveling salesman problem [6] etc.

ACO is based on two core concepts:

- incremental solution construction by virtual agents (called ants) through biased random exploration,
- stigmergy by updating the pheromone values, which are accessible to all the agents.

The basic skeleton of the ACO meta heuristic is presented in Algorithm 2.1[26].

Algorithm 2.1 ACOmetaHeursitic

```
1: initialize algorithmic parameters
2: initialize pheromones
3: while stopping criteria is not satisfied do
4:     construct solution
5:     (optional)perform local search
6:     update pheromones
```

After initializing the algorithmic parameters and pheromones, the actual search process begins. The search is carried out until a given stopping criteria is satisfied. The following actions are performed during each iteration of the search:

- Construct solutions:

A number of ants are employed to parallelly construct tentative solutions using probabilistic rules. The probabilistic rules used in this step are functions of pheromone values. These tentative solutions constructed by the ants are in fact feasible solutions of the problem.

- Perform local search:

  A problem specific optional local search procedure may be applied to farther improve the tentative solutions constructed by the ants.

- Update pheromone :

  This phase consists of two steps, pheromone evaporation, and pheromone depositing. Pheromone evaporation is performed first to mimic the natural phenomena of pheromone reduction. In pheromone depositing step, a sub-set of the solutions constructed by the ants in the previous phase are selected as sufficiently good quality solutions. Only the ants generating these solutions are allowed to deposit pheromone. The amount of pheromone deposited by each ant is a function of the quality of the solution generated by that ant. In this way, the results of one iteration guides the explorations carried out in next iteration.

The modeling of a problem for solving with ACO and different steps of the ACO meta heuristic are described next.

2.6.1. Problem Modeling and Algorithmic Parameters

A problem is modeled as a graph with a finite set of positions or states, each being a vertex in the graph. An edge exists from the positions i to j if selecting j is allowed after selecting i. Selecting the edge (i, j) is synonymous to transitioning from position i to j. The cost of transitioning from a position i to j is denoted by

η_{ij}, and is called heuristic information. The heuristic information is specific to a problem, and is assumed to be known a priori.

The following algorithmic parameters are provided as inputs, all real valued:

- The pheromone importance parameter,α
- The heuristic importance parameter, β
- The pheromone evaporation constant, $\rho \in [0.0, 1.0]$

The algorithmic parameters provide means of controlling the behavior of the algorithm. Typical values for solving TSP problem are shown in Table 2.1.

2.6.2. Pheromone Representation and Initialization

Pheromone is represented by τ, a matrix of real numbers. Each entry τ_{ij} in the pheromone matrix denotes the amount of pheromone present on the edge (i, j). The amount of pheromone on an edge (i, j) denotes the attractiveness of selecting j after selecting i in the solution. At the beginning of the search, pheromone values are initialized with a pre-specified amount τ_0. The value of τ_0 requires careful selection, because if it is too low, then the search is quickly biased by the first solutions generated by the ants, which in general leads to premature optimization. On the other hand, if the initial pheromone values are too high, then many iterations are lost waiting until pheromone evaporation reduces enough pheromone values, so that pheromone added by ants can influence the search.

Ideally, the initial pheromone value should be low enough that pheromone added due to the new solutions generated by ants can have influence on the search, but high enough to provide equal preference to all edges during the initial phase of the search. Following this rationale, it has been suggested [26] that calculating a greedy solution S_g of the problem and the cost of such a solution C_g can aid in pheromone initialization using the following formula:

$$\tau_0 = \frac{1}{C_g} \tag{2.3}$$

2.6.3. Solution Construction

During the Solution Construction phase, m number of ants are employed to construct tentative solution. The value of m can be a constant e.g. 1, 2, 10 or relative to the problem size e.g. for TSP problem, m being equal to the number of cities is suggested[26].

Each ant constructs a solution in an incremental manner by biased random exploration of the problem graph. Starting from an initial position, each leg of the exploration consist of a transition from position i to position j until a solution is considered complete. The probability P_{ij} of selecting position j after position i is calculated by the following rule.

$$P_{ij} = \frac{[\eta_{ij}]^{\alpha} \times [\tau_{ij}]^{\beta}}{\sum_{j \in N(i)} [\eta_{ij}]^{\alpha} \times [\tau_{ij}]^{\beta}} \tag{2.4}$$

Here, $N(i)$ denotes the set of positions adjacent to i which are unvisited by this ant. The criteria for determining the completeness of a solution, and definition of $N(i)$, the neighborhood of a position i vary from one problem to another.

2.6.4. Local Search

As a meta heuristic, the ACO approach is tasked with guiding a search procedure through the problem space[63]. Different local search techniques have been reported in scientific literature for solving different problems. The 2-opt local search procedure has been suggested for solving TSP[26]. An iterative local search technique for solving the bin packing and stock cutting problem has been presented in [51]. Local search techniques named 1-shift and 2.5-opt-EEais are suggested for solving the probabilistic traveling salesman problem[6]. Although the use of a local search procedure is considered optional, coupling the ACO with a local search has been observed to improve the result[26].

2.6.5. Pheromone Update

Pheromone update consists of two steps: *pheromone evaporation*, and *pheromone depositing*. In the pheromone evaporation step, the natural phenomena of pheromone reduction due to evaporation is simulated. This is carried out by updating all the entries in the pheromone matrix τ using the formula:

$$\tau_{ij} = (1 - \rho)\tau_{ij} \tag{2.5}$$

In the pheromone depositing step, a sub-set of solutions are selected, out of the m solutions produced by the ants during exploration phase, according to a preference rule. The preference rules are one of the main differences between different versions of the ACO approach. Only those ants who were responsible for generating these preferred solutions are allowed to add pheromone. Let a preferred solution be S_p and its cost be C_p, then the respective ant deposits pheromone according tho the following formula:

$$\tau_{ij} = \begin{cases} \tau_{ij} + \frac{1}{C_p} & \forall (i,j) \in S_p \\ \tau_{ij} & otherwise \end{cases} \tag{2.6}$$

In this way, ants generating solutions with relatively higher quality are allowed to guide the subsequent stages of the search by depositing pheromone. This in turn, results in gradual improvement of the overall quality of solutions generated by the ants in each iteration.

The value of ρ determines how quickly or slowly the search should converge. For large value of ρ, the pheromones reduces rapidly, causing the newly added pheromone in the beginning of the search to heavily bias the exploration in subsequent iterations. This causes a subset of edges being used repeatedly in solution construction and a pheromone accumulation takes place in those edges. As a result, the search converges quickly to generating solutions using that subset of edges. On the other hand, for smaller values of ρ, the pheromone reduces slowly. This allows the ants to explore a wider range of edges in the initial phase. Eventually a significant amount of pheromone accumulates on a subset of edges that are frequently included in good quality solutions. Experimental results show that lower values of ρ result in better quality results compared to using higher values of ρ[26], because in the latter case the algorithm converges prematurely.

2.6.6. Variations of ACO

The earliest and most rudimentary ACO approach, the *Ant System (AS)*[21, 25] follows the skeleton algorithm (Algorithm 2.1). The AS was introduced as a meta heuristic for solving the TSP [25]. It was initially found to be promising, but did not fare well in comparison to the state-of-the-art TSP algorithms. This limitation was overcome later by introducing several extensions of the AS, with increasingly improved performance. Two prominent extensions of the AS are the *Elitist Ant System (EAS)*[21, 24, 23], and the Max Min Ant System(MMAS)[75]. The main difference between different ACO approaches lies in the pheromone initialization and pheromone update rules. Apart from that, the preference of values of algorithmic parameters also vary. The extension of AS are described next:

Elitist Ant System(EAS):

The core idea of EAS is to provide strong additional reinforcement to the edges included in the best solution found since the start of the algorithm[21, 25, 24]. This is carried out by depositing additional pheromone on the edges included in the global best solution at the end of each iteration. Let S_B denote the global best solution found since the beginning of the algorithm, and C_B be the cost of this solution, then additional pheromone is added to the edges included in S_B according to the following formula:

$$\tau_{ij} \leftarrow \tau_{ij} + \frac{e}{C_B} \; \forall (i,j) \in S_B \tag{2.7}$$

where e is an additional algorithmic parameter introduced in the EAS. Experimental results presented in [21, 25, 24] suggest that selecting an appropriate value for e can improve the quality of result as well as reduce the required number of iteration. Since this additional step is the only difference between AS and EAS, the EAS approach is not discussed in further details.

Max Min Ant System(MMAS):

The MMAS is the most sophisticated version of AS. The following four major modification are incorporated in this approach[75, 74]:

- Strong Restriction on Pheromone Deposition

  Pheromone is added based on either the global best solution S_B or the iteration best solution S_b only. Unlike other variations of the AS, not all the ants employed in an iteration are allowed to deposit pheromone. Either S_B or S_b is chosen using a probability rule in which, the chance of S_B being selected increases with each iteration.

- Pheromone Limits

  Because of the first modification, there is a possibility of large accumulation of pheromone on a small subset of edges. This effect is countered by limiting the pheromone values within the a dynamic interval $[\tau_{min}, \tau_{max}]$. In the beginning of the algorithm the values are set by $\tau_{max} = \tau_0$ and $\tau_{min} = \frac{\tau_{max}}{a}$ where, τ_0 is calculated from Equation 2.3, and a is an additional algorithmic parameter. Afterwards, τ_{min}, and τ_{max} are updated at the end of each iteration according to the following rule:

$$\tau_{max} = \frac{1}{\rho C_B} \tag{2.8}$$

$$\tau_{min} = \frac{\tau_{max}}{a} \tag{2.9}$$

  where, C_B is the cost of the global best solution S_B. The value of a should vary from one problem to another. For the TSP, Setting $a = \frac{\sqrt[n]{0.05}\times(avg-1)}{1-\sqrt[n]{0.05}}$

where, n is the input size and avg is the average number of choices available to an ant during the construction of a solution has been suggested in [75].

- Encouragement for Initial Exploration

  A small value of ρ is set, to encourage exploration at the start of the search. For the TSP, $\rho = 0.02$ is suggested[75].

- Pheromone Reinitialization

  Pheromone trails are reinitialized to current value of τ_{max} each time the system approaches stagnation or when no improved tour has been generated for a certain number of consecutive iterations.

The suggested values[26] of algorithmic parameters for solving the TSP using different ACO approaches are summarized in Table 2.1.

ACO Algorithm	α	β	ρ	m	τ_0
AS	1	$[2, 5]$	0.50	n	$\frac{m}{C_g}$
EAS	1	$[2, 5]$	0.50	n	$\frac{n+m}{\rho C_g}$
MMAS	1	$[2, 5]$	0.02	n	$\frac{1}{\rho C_g}$

Table 2.1.: Suggested values of algorithmic parameter for solving the TSP using different ACO approaches.

2.7. Extension of ACO for Continuous Domain

The Ant Colony Optimization (ACO) meta heuristic was originally meant to be used for combinatorial optimization problems. An extension of ACO, called the Continuous Ant Colony Optimization (ACO$_{\mathbb{R}}$), was proposed later [73] to handle optimization problems in continuous domains as well. This section presents an overview of the ACO$_{\mathbb{R}}$ followed by description of the actual algorithm.

In the large group of algorithms for solving continuous optimization problems, $\text{ACO}_{\mathbb{R}}$ has been classified as an Evolutionary Algorithm (EA) due to the similarity between $\text{ACO}_{\mathbb{R}}$ and many other EAs[73]. Some other EAs are presented in [64], [37]. A comparative study of these algorithms along with other EAs can be found in [46]. According to the authors of [73], these different algorithms share a common trait with $\text{ACO}_{\mathbb{R}}$: learning and modeling explicitly probability distributions.

2.7.1. The $\text{ACO}_{\mathbb{R}}$ Algorithm

The following actions are carried out iteratively in the $\text{ACO}_{\mathbb{R}}$ algorithm until a predefined stopping criteria is met.

- A number of ants are employed to construct tentative solutions by taking random decisions biased by a collection of existing solution instances. An optional local search may be performed on the solutions.
- The collection of solutions are updated by replacing low quality solutions with high quality solutions (if any) from the newly created solutions by the ants.

A Continuous Optimization Problem (CnOP) defined for solving with $\text{ACO}_{\mathbb{R}}$ as $Q = (\mathbf{S}, \Omega, f)$ where:

- $\mathbf{S}$ is a search space defined over a set of n continuous decision variables
- Ω is a set of constraints
- $f : \mathbf{S} \to \mathbb{R}_0^+$ is the objective function to be minimized

A solution instance S is defined as a set of continuous variables $X_i, i = 1, \ldots, n$.

The central idea behind the ACO meta heuristic is the incremental construction of solutions based on the biased probabilistic choice of solution components. This is reflected in Algorithm 2.1. The $\text{ACO}_{\mathbb{R}}$ follows the same idea and algorithmic structure. However, the pheromone representation, and solution construction are are carried out differently.

Pheromone Representation

In ACO$_\mathbb{R}$ pheromone information is stored in the Pheromone table T. Each entry in T contains a solution instance. Unlike its discrete domain counterpart, the solutions are not discarded after depositing pheromone. The solutions in T are ordered according to their quality, i.e. based on the value of the objective function. The pheromone table is illustrated in Table 2.2. While the pheromone matrix τ plays the role of implicit memory during the execution of ACO, the pheromone table T in ACO$_\mathbb{R}$ serves as explicit memory.

s_1	s_1^1	s_1^2	$\cdots$	s_1^i	$\cdots$	s_1^n	$f(s_1)$	ω_1
s_2	s_2^1	s_2^2	$\cdots$	s_2^i	$\cdots$	s_2^n	$f(s_2)$	ω_2
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
s_l	s_l^1	s_l^2	$\cdots$	s_l^i	$\cdots$	s_l^n	$f(s_l)$	ω_l
	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
s_k	s_k^1	s_k^2	$\cdots$	s_k^i	$\cdots$	s_k^n	$f(s_k)$	ω_k

Table 2.2.: Pheromone table for ACO$_R$, the entries are sorted according to solution quality $f(s)$

The size of T, denoted by k, refers to the number of entries to be stored in T and is provided as an algorithmic parameter. In the beginning of the algorithm, T is initialized by populating with uniform-random samples.

Solution Construction

The number of ants employed during solution construction, m is an algorithmic parameter, and is provided as an input. During each iteration of the algorithm, each ant constructs a tentative solution. The solution construction begins with probabilistically selecting a solution entry from T. The probability of selecting the jth solution is given by :

$$p_j = \frac{\omega_j}{\sum_{r=1}^{k} \omega_r} \tag{2.10}$$

where, ω_j denotes the weight of the jth solution, given by:

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-1(j-1)^2}{2q^2k^2}} \tag{2.11}$$

where, q is an algorithmic parameter. An increasingly wider range of solutions become likely to be selected as the value of q rises. Small values of q give more weight to a narrow range of solutions concentrated near the best entries.

Once a solution s_l is selected, the ant samples the neighborhood of each decision variable in s_l. For the ith decision variable in s_l, denoted by s_l^i, the sampling is performed using a probability density function. The use of a probability density function, which is a continuous function, is a major difference of ACO$_{\mathbb{R}}$ from ACO where a discrete probability distribution function (the Equation 2.4, for example) is used during solution construction. Any real valued positive function $P(x)$ can be used as the probability distribution function as long as the following criteria is satisfied:

$$\int_{-\infty}^{\infty} P(x)dx = 1 \tag{2.12}$$

However, the authors of [73] suggest using Gaussian function as the probability distribution function for sampling according to the following formula:

$$P(x) = g(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.13}$$

where, the parameters μ and σ are defined for each s_i^j as follows:

$$\mu = s_i^j \tag{2.14}$$

$$\sigma = \xi \sum_{r=1}^{k} \frac{|s_r^i - s_j^i|}{k-1} \tag{2.15}$$

here, ξ is an algorithmic parameter. Higher values of ξ allows the resulting samples being taken from a wider area, as a result the algorithm explores more and converges slowly. Lower values of ξ on the other hand, narrow down the sampling, resulting in fast convergence with less exploration. The role of ξ is similar to the pheromone evaporation constant ρ in ACO in this regard [73].

Pheromone Update

Each of the m ants employed in an iteration constructs a solution in this way. At the end of the solution construction phase, the new m solutions are added to the pheromone table T, which contained k entries prior to this action. From the resulting $k + m$ entries in T, the worst m entries are removed. In this way, the collection of solutions guiding the search is refined with each iteration.

3. Proposed Approach

A heuristic to jointly optimize the relay node placement and the trajectory of a mobile data collector in a 3-tier wireless sensor network (RNPT problem) is presented in this chapter. The proposed approach operates in a 2-phase process. Taking the coordinates of a set of sensors as input, a list of coordinates for placing relays in the order meant to be visited by the mobile data collector is produced in the first phase using an ant colony optimization approach. This ordered list is in fact a feasible trajectory for the mobile data collector. This feasible trajectory if farther optimized in the second phase. A deterministic algorithm, and a continuous ant colony optimization approach is presented in this chapter for further optimizing the feasible trajectory.

3.1. The Network Model

A three-tiered wireless sensor network (Figure 3.1) is considered, where the lowest tier comprises of a set of sensors. The sensors are assumed to be deployed to ensure appropriate coverage of the sensing field. The sensors are organized in clusters, where a relay node acts the *cluster-head* for each cluster. The middle tier consists of the relays. Each relay is equipped with higher capacity CPU, memory, and power source, and is capable of collecting, buffering and transmitting the data collected from sensors over a larger distance. The top tier consists of a mobile data collector which is a base station mounted on a vehicle. The mobile data collector operates without any power or memory constraints. The duty of the mobile data collector is to traverse a pre-calculated trajectory to visit each of the relays in a predetermined sequence and collect the data buffered in the relays.

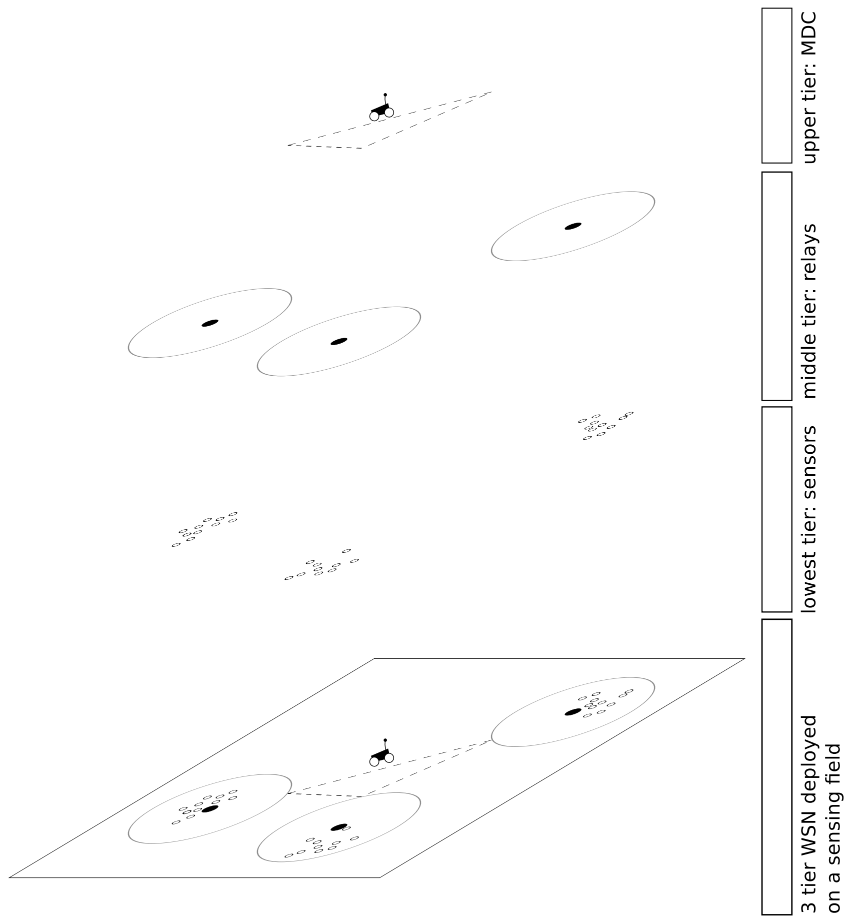


Figure 3.1.: A 3-tier WSN.

3.2. Problem Formulation

It is assumed that the number of sensors and their positions in a flat 2-dimensional sensing field are known. There are no obstacles in the field. Here $r(R)$ denotes the communication range of the sensors(relays). The data from a sensor can be collected by a relay as long as it lies within the region of influence of the sensor, which is a disk of radius r centered at the sensor. Two or more sensors can have overlapping regions of influence. Buffered data from a relay can be downloaded to the MDC as long as it gets sufficiently close to the relay, i.e. within R distance from it. The regions of influences of the sensors and the relays are illustrated in Figure 3.2. Since relays are equipped with higher capacity batteries, R is greater than r.

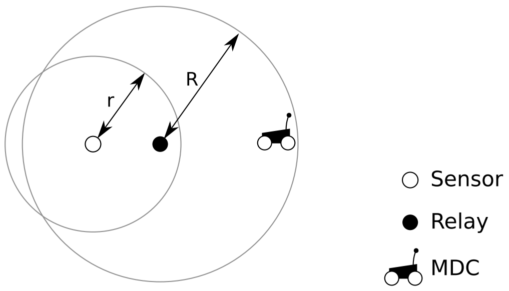


Figure 3.2.: Regions of influences of the sensors and the relays.

The objective is to calculate a placement of relay nodes and a trajectory for the MDC based on the said placement such that the number of relays and the length of the trajectory are optimized.

The proposed solution approaches the problem in a two-phase process. In the first phase, given the number and the locations of the sensors in a sensing field, an ordered list of locations for placing relays is produced such that, every sensor is covered by at least one relay while the number of relays and the total distance traveled by the MDC when they are visited in the given order is optimized. This is carried out by an ACO approach, presented in section 3.3. Since the resulting trajectory found in the first phase is a feasible solution of the problem, henceforth it is referred to as a *feasible trajectory.*

Such a trajectory determines the sequence in which the MDC travels from one relay to another in order to download the buffered data. In that regard, the resulting trajectory presents a TSP solution. It has been mentioned earlier in section 2.3 that, a TSPN solution results in a shorter trajectory compared to the corresponding TSP. Therefore, a feasible trajectory found from the first phase can be further refined by calculating a TSPN-like tour where the neighborhoods are disks centered at each relay. This refinement is carried out in the second phase. Two alternatives are presented for refining a feasible trajectory: a deterministic heuristic in section 3.5, and an $\text{ACO}_{\mathbb{R}}$ approach in section 3.6. The scenario presented in Figure 3.3 shows a sensing field containing 8 sensors labeled as s_1, s_2, ... s_8. This scenario is referred to as the *example scenario* in the remainder of this chapter, and is used to illustrate different stages of the proposed approach.

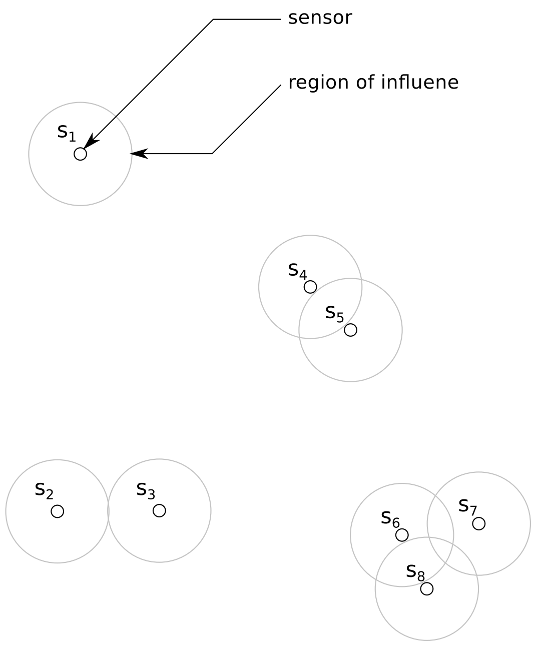


Figure 3.3.: An example scenario.

3.3. Relay Placement and Trajectory Calculation

An ACO approach to RNPT problem is presented in this section. Before the actual ACO algorithm can begin, a preprocessing step for calculating the potential relay locations is performed. Different components of the proposed ACO approach are described next. The result of the proposed approach is a trajectory expressed as a list, containing a subset of the potential relay locations.

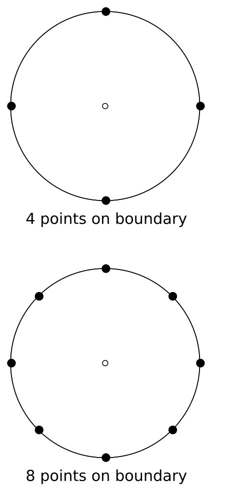
3.3.1. Calculation of Potential Relay Locations

From a given set of sensors, the first step towards the solution is to calculate $\mathcal{R}$, a set of potential relay locations. Our proposed heuristic for selecting a set of potential relay locations begins by initializing $\mathcal{R}$ as an empty set.

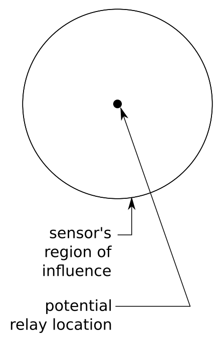
For each pair of sensors whose regions of influence overlap, the points of intersection between the circles with radius r and centered at the sensors are added to the set of potential relay locations. Note: there can be one or two such points depending on whether the circles are touching or intersecting. Such locations are referred to as $class - I$ locations henceforth. In Figure 3.5, the locations labeled r_2, r_3, $\dots$ r_{10} are such locations.

The sensors whose regions of influence do not overlap with those of any other sensors are not yet covered by any potential relay location. Such senors are referred to as *disconnected sensors* henceforth. For each disconnected sensor, a number of evenly spaced points on the boundary of the sensor's region of influence are added to $\mathcal{R}$. These locations are also classified as $class - I$ locations. The concept is illustrated in Figure 3.4(a) for 4 and 8 points. A limitation of this selection criteria is that the size of $\mathcal{R}$ grows by factor of 4, or 8 with the number of disconnected sensors.

An alternative strategy for covering the disconnected sensors, is to add the location



(a) Evenly spaced points on the boundary of a sensor's region of influence.



(b) Selecting the location of the sensor as a potential relay location.

Figure 3.4.: Different strategies for selecting potential relay locations for disconnected sensors.

of the sensor itself to $\mathcal{R}$. This strategy ensures that the size of $\mathcal{R}$ would not exceed the number of sensors. These locations however, would need special treatment during calculation of trajectory. Therefore, they are classified as $class-II$ locations, to distinguish from the $class-I$ locations. This concept is illustrated in Figure 3.4(b). The location labeled r_1 in Figure 3.5 is such a location. Results of experimentation with three different strategies for choosing potential relay locations i.e., selecting the centre point, or evenly spaced 4 points, or 8 points on the boundary are presented in Chapter 4.

Let the number of locations be denoted by N. An $N \times N$ matrix D is maintained such that the ith row and column in D corresponds to the ith potential relay location. Each entry d_{ij} in D represents the euclidean distance from location i to location j. In order to speed up the execution, distance values are looked up from D instead

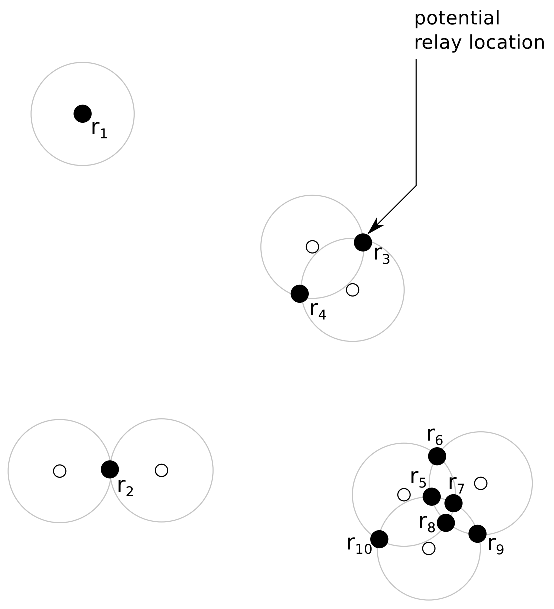


Figure 3.5.: Potential relay locations in the example scenario (labels of sensors are omitted for simplicity).

of calculating in all subsequent stages. Using an ACO approach, a subset from the set of potential relay locations are chosen in an ordered list such that the following design goals are achieved:

- Every sensor is covered by at least one relay in the list,
- The number of relays are optimized,
- The total traveling distance, when the relays are visited in that order, is minimized.

3.3.2. ACO Approach for Jointly Solving Relay Placement and Trajectory Calculation

The proposed ACO meta-heuristic for jointly solving the relay placement and trajectory problem is presented in this section. Since the joint problem of relay node placement and trajectory calculation has much in common with the TSP problem, the proposed ACO approach follows the framework for solving the TSP, as presented in [75, 26]. Each iteration of the proposed approach consists of two steps, the solution construction, and the pheromone updating step. The following tasks are carried out during each iteration:

- Solution Construction:

  a pre-specified number of ants are employed to construct tentative solutions. Problem specific heuristic information and the existing pheromone trace is consulted during the construction of a solution by each ant. The solution constructed by each ant is refined by a local search procedure, then stored in the memory.

- Pheromone Updating:

  the pheromone trail values are updated based on the solutions generated in the exploration phase. A stagnation detection method, described in Equation 3.3.8, is also used to prevent the meta-heuristic from narrowing down to a possibly local optima. Pheromone trails are reinitialized upon detection of stagnation.

The result of ACO approach is a trajectory for the MDC expressed as a list of points. Such a trajectory found from the example scenario is illustrated in Figure 3.6. The potential relay locations chosen are r_1, r_2, r_5, and r_4 , and they are intended to be visited by the MDC in that order. Labels of sensors are omitted for the sake of simplicity. Different components of the proposed ACO approach is described in detail in the following sub-sections.

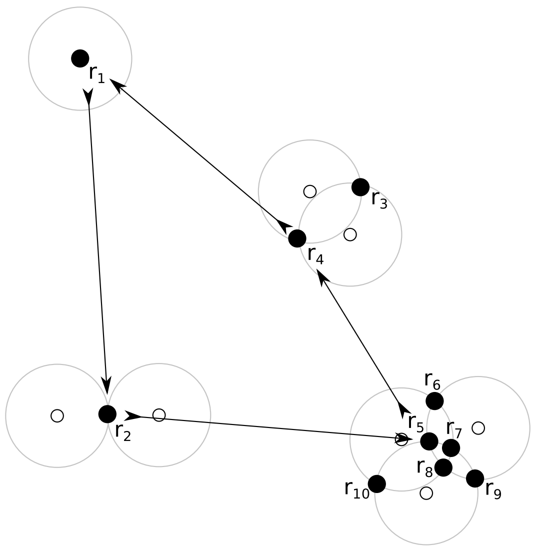


Figure 3.6.: Outcome of the proposed ACO approach.

3.3.3. Solution Representation & Cost Metric

A solution instance essentially represents a trajectory for the MDC. For the remainder of this thesis, a solution instance is assumed to be represented as a list of locations. Each entry in the list is a 2-dimensional point in the Euclidean space. For each point u in the list, $next(u)$, and $previous(u)$ denotes the points to be visited after, and before visiting u respectively. Since this is a closed trajectory, if u is the first entry, then $previous(u)$ refers to the last entry of the list, and if u is the last entry, then $next(u)$ refers to the first entry of the list.

Since the solution is a list, $index(u, S)$ denotes the index of a particular point u in a solution instance S. The number of entries in a solution S is denoted by l_s. The cost of a solution can be measured by two different metrics C_s^1, and C_s^2 as described next.

The total length of the trajectory suggested by a solution instance is an important component of the cost. From the WSN perspective, the number of relays used in a solution instance is also of significant importance, as mentioned in subsection 2.1.4. The metric C_s^1 estimates the cost of a solution s taking both of these factors into account using the formula:

$$C_s^1 = l_s \sum_{u \in s} d_{u,next(u)} \tag{3.1}$$

A simplified metric C_s^2 estimates the cost as the total trajectory length, ignoring the contribution of the number of relays used, using the formula:

$$C_s^2 = \sum_{u \in s} d_{u,next(u)} \tag{3.2}$$

Results of experiments performed with both metrics are presented in the Chapter 4 of this thesis.

3.3.4. Heuristic Information

The heuristic information is a measurement of the attractiveness of a particular move during the solution construction by an ant. In the context of our problem, making a move is synonymous to adding a particular location to the solution being constructed by that ant.

Two factors contribute to the relative attractiveness of a move: distance traveled to make that move, and the number of uncovered sensors that can be covered by making this particular move. The attractiveness being inversely proportional to the distance traveled to make a move, and directly proportional to the number of uncovered sensor that can be covered by making that move. A metric for the heuristic information taking both of the factors into account is calculated using the formula:

$$\eta_{ij}^{1} = \frac{u(j)}{d_{ij}} \tag{3.3}$$

where, η_{ij}^{1} denotes the attractiveness of selecting location j after selecting location i, and $u(j)$ denotes the number of uncovered sensors that can be covered by including j.

An alternative metric which takes into account only the distance traveled in making that move is calculated using the formula:

$$\eta_{ij}^{2} = \frac{1}{d_{ij}} \tag{3.4}$$

Where η_{ij}^{2} is the attractiveness of selecting the jth location after selecting the ith location without considering the contribution of the number of uncovered sensors covered by making that move. Results of experiments preformed using the two different heuristic information are presented in Chapter 4.

3.3.5. Pheromone representation and Initialization

Pheromone is represented by an $N \times N$ matrix τ. Each entry τ_{ij} in the pheromone matrix denotes the intensity of pheromone trace from the ith location to the jth location. Adapting the framework presented for solving the TSP [26, 75], the pheromone matrix is initialized with $\tau_0 = \frac{1}{C_g}$, where C_g is the cost of an initial greedy solution. The initial greedy solution is found by calculating a TSP tour using the Nearest-

Neighbor(NN) heuristic(Algorithm 3.2), of a subset of potential relay locations calculated using a greedy cover algorithm (Algorithm 3.1).

Algorithm 3.1 GreedyCover

```
Input: $\mathcal{R}$, set of potential relay locations
    $\mathcal{S}$, set of sensors
Output: $\mathcal{T}$, a subset of $\mathcal{R}$ that covers all sensors in $\mathcal{S}$
 1: $\mathcal{T} \leftarrow \emptyset$
 2: while $\mathcal{S} \neq \emptyset$ do
 3:     $r \leftarrow$the relay in $\mathcal{R}$ that covers the most number of sensors in $\mathcal{S}$
 4:     $\mathcal{T} \leftarrow \mathcal{T} \cup r$
 5:     $\mathcal{R} \leftarrow \mathcal{R} - r$
 6:     remove the sensors covered by $r$ from $\mathcal{S}$
 7:     removes the relays in $\mathcal{R}$ which do not cover any sensor in $\mathcal{S}$
```

Algorithm 3.2 NN-TSP

```
Input: $\mathcal{T}$, a set of locations
Output: $\mathcal{P}$, a trajectory expressed as a list locations
 1: $\mathcal{P} \leftarrow \emptyset$
 2: $r \leftarrow$first element in $\mathcal{T}$
 3: add($\mathcal{P}, r$)
 4: $\mathcal{T} \leftarrow \mathcal{T} - r$
 5: while $\mathcal{T} \neq \emptyset$ do
 6:     $s \leftarrow$the nearest potential relay location from $r$ which belongs to $\mathcal{T}$
 7:     $r \leftarrow s$
 8:     add($\mathcal{P}, r$)
 9:     $\mathcal{T} \leftarrow \mathcal{T} - r$
```

3.3.6. Solution Construction

Each individual ant employed during an iteration incrementally constructs a tentative solution. When the number of ants is the same as the number of potential relay locations, one ant starts from each location. In case the number of ants is less than the number of potential relay locations, a sub-set of locations are chosen using uniform random sampling, without replacement; an ant starts from each location in this sub-set[26]. This location is called the *starting-location* of an ant.

Each ant maintains two sets of potential relay locations:

- *Set of visited-locations, $\mathcal{U}$*:

  The set of potential relay locations that have been included in the solution constructed by this ant; initialized as a set containing only the starting-location of the ant.

- *Set of unvisited-locations, $\mathcal{V}$:*

  The set of potential relay locations not yet included in the solution constructed by this ant; initialized to contain all the potential relay locations except the starting-location of the ant.

It has been shown that maintaining a list of nearby location can speed up the solution construction process [26, 22]. A dynamic list called the *candidate-list* is maintained for each location. The candidate list of the ith potential relay location, denoted by $\mathcal{N}(i)$, holds the most attractive potential relay locations in $\mathcal{U}$ reachable from the ith location, sorted in descending order of attractiveness. The size of the candidate-list can be a constant (i.e., 2, 4, 10 etc.) or relative to the number of locations (i.e., $N/4$).

The *current-location* of an ant refers to the most recent potential relay location included in the solution being constructed by that ant. In the beginning of solution construction by an ant, its starting-location is included in the solution, and is set as its current-location. The incremental construction of a tentative solution now begins. The following actions are repeated until the tentative solution is considered completed:

An ant located at the ith location probabilistically selects the next location to add to the solution using the formula which was introduced earlier in section 2.6:

$$P_{ij} = \frac{[\eta_{ij}]^{\alpha} \times [\tau_{ij}]^{\beta}}{\sum_{j \in N(i)} [\eta_{ij}]^{\alpha} \times [\tau_{ij}]^{\beta}} \tag{3.5}$$

where α,β are algorithmic parameters and $N(i)$ is candidate list of i.

Upon selection of the next location j, the following actions are performed:

- j is added to the solution,
- j is added to $\mathcal{V}$,
- j is removed from $\mathcal{U}$,
- j is set as the current-location of the ant,
- redundant locations in $\mathcal{U}$ are removed,
- candidate lists are updated.

When a decision is made to place a relay at a potential relay location, it is possible that some of the remaining potential relay locations will be redundant. If all sensors covered by a relay placed at a potential relay location l is already covered by the locations already included in the solution, then l can be removed from the set of potential relay locations to be considered in constructing the solution. Following this rationale, all such redundant locations in $\mathcal{U}$ are removed.

The removal of the jth potential relay location from $\mathcal{U}$ makes it unreachable from the remaining members of $\mathcal{U}$. As a result, the candidate lists of the remaining members of $\mathcal{U}$ needs updating. In addition to that, the locations which became redundant due to the inclusion of j in the tentative solution should also be removed from the candidate list of any member of $\mathcal{U}$.

The solution construction by an ant is considered completed when all the sensors are covered by at least one potential relay location in $\mathcal{V}$. All the solutions constructed during an iteration is stored in the memory and is used during the pheromone update phase. The solution construction procedure is presented in Algorithm 3.3.

Algorithm 3.3 SolutionConstruction

```
Input: $t$, starting location of the ant
    $\tau$, pheromone matrix
    $\mathcal{R}$, set of potential relay locations
Output: $S$, a solution instance
 1: $\mathcal{U} \leftarrow \{t\}$
 2: $\mathcal{V} \leftarrow \mathcal{R} - \{t\}$
 3: $c \leftarrow t$
 4: $S \leftarrow \emptyset$
 5: for all $u \in \mathcal{U}$ do
 6:     initilize candidate list $\mathcal{N}(i)$
 7: while $S$ is not a complete solution do
 8:     select $j$ using the probability rule in Equation 3.5
 9:     $add(S, j)$
10:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{j\}$
11:     $\mathcal{U} \leftarrow \mathcal{U} - \{j\}$
12:     $c \leftarrow j$
13:     remove redundant locations from $\mathcal{U}$
14:     for all $u \in \mathcal{U}$ do
15:         update the candidate list $\mathcal{N}(u)$
```

3.3.7. Local search

A local search procedure is applied to further optimize the solution constructed by each ant. Referring back to subsection 2.6.4, several local search procedures are found in the literature, of which 2-opt is commonly known.

Given a tour as a sequence of nodes, the 2-opt operation performs the following tasks:

- selects a pair of edges $(u_{1,}u_2)$ and (v_1, v_2), appearing in this order,
- replaces them by a new pair of edges $(u_{1,}v_1)$, (u_2, v_2),
- reverses the direction of edges appearing between $v_1$and u_2.

This results in a new tour with possibly different tour cost. The 2-opt local search is elaborated with an example in Figure 3.7. In the given scenario a possible tour $acbdef$ can be improved by switching the edge (a, c) with (b, d). The resulting tour $abcdef$ has shorter length than the previous one.

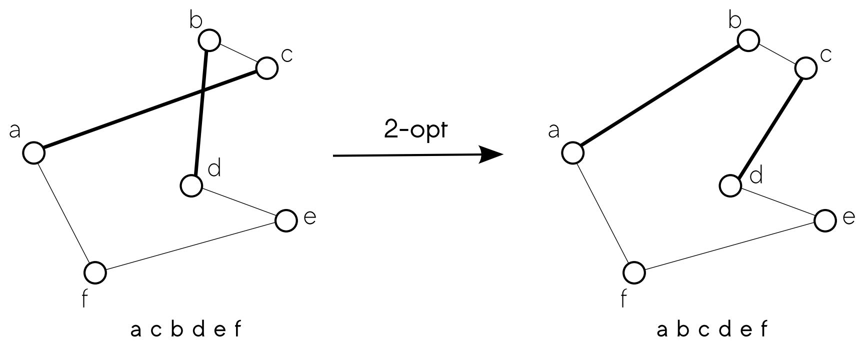


Figure 3.7.: 2-opt local search.

For each solution generated during the solution construction phase, the 2-opt operation leading to maximum cost reduction is determined and performed. This is similar to a steepest descent search[90].

At the end of the iterative part, once the stopping criteria has been satisfied, the global-best solution is refined by performing a sequence of 2-opt operations leading to highest cost reduction until no more 2-opt operations can be performed. This greedy optimization has been observed to considerably improve the quality of the result.

3.3.8. Components of the MMAS

Among several variations of ACO, experiments were performed with the Ant System(AS), the Elitist Ant System(EAS) and, the Max Min Ant System(MMAS). The experimental results are presented in Chapter 4. The basic structure of ACO is followed in all three above mentioned variations. The modifications specific to the EAS has been described earlier in Chapter 2. Only the MMAS specific modifications are described here. The Algorithm 3.4 shows the basic structure of the MMAS. Compared to AS, or EAS, the modifications in MMAS can be seen in: pheromone update, enforcing pheromone limits, stagnation detection and pheromone reinitialization.

Algorithm 3.4 ACO-meta-heursitic

```
1: initialize algorithmic parameters
2: initialize pheromone values
3: while stopping criteria is not satisfied do
4:     construct solutions
5:     update pheromone values
6:     if stagnation is detected then
7:         reinitialize pheromone values
```

Pheromone Update

Pheromone update comprises of the following two steps:

Step-1: The first step is referred to as the pheromone evaporation step(subsection 2.6.5). The natural phenomena of pheromone intensity reduction by evaporation is mimicked in this step. It is performed by reducing the value in each entry of the pheromone matrix by factor of $(1-\rho)$ where ρ is the evaporation rate. The following formula is used to perform pheromone evaporation on each entry τ_{ij} of the pheromone matrix.

$$\tau_{ij} = (1-\rho)\tau_{ij} \tag{3.6}$$

Step-2: In the second step, referred to as the pheromone depositing step(subsection 2.6.5), pheromone is added to a subset of pheromone trails. A sub-set of solutions constructed during the previous phase is selected as preferred solutions for depositing pheromone. For small problem instances (i.e. $N \leq 200$) only the iteration-best solution is chosen in every iteration and the ant responsible for generating this solution is eligible for depositing pheromone [26]. For larger problem instances, the iteration-best and the global-best solution is selected alternatively, with the global-best solution being selected with increasing probability. The probability p of the global-best solution being selected is calculated using:

$$p = \frac{1}{1 + \log(k)} \tag{3.7}$$

Here, k denotes the iteration number. According to the principals of MMAS, only the ant responsible for generating the selected solution is eligible for depositing pheromone [26, 75]. Let s_{pl} be the selected solution, C_{pl} be the cost of the tour, and the edge (i, j) denote that j is selected after i in the solution. Pheromone is deposited using the following formula:

$$\tau_{ij} = \begin{cases} \tau_{ij} + \frac{1}{C_{pl}} & if\ i \to j \in s_{pl} \\ \tau_{ij} & otherwise \end{cases} \tag{3.8}$$

The pheromone evaporation rate ρ is an important algorithmic parameter. The extent of exploration performed by the ACO meta-heuristic is controlled by specifying the value of ρ. For larger values of ρ, the pheromone traces are evaporated quickly, leading to narrowing down of the search into a promising avenue. This involves the risk of premature termination of the optimization process. For smaller values of ρ on the other hand, pheromone traces reduce slowly, allowing the ants to explore more during each iteration. The search converges slowly, but with more chances of finding good solutions. Using $\rho \sim 0.02$ has been suggested for MMAS [75].

The selection of solutions for depositing pheromone in one iteration affects the search procedure by influencing the explorations in subsequent iterations. Therefore, only relatively good quality solutions should be used for depositing pheromone. Selecting the global best solution every time causes the search to quickly converge in the neighborhood of the global best solution. This premature optimization is avoided by using the solution selection criteria described in Step 2. The rationale behind the selection criteria is that the search should perform more exploration in the beginning, and is expected to gradually narrow down to a relatively promising neighborhood within the search space. By selecting the iteration-best solution in early iteration, or always in the case of small problem instances, exploration is encouraged. By updating the pheromone trails based on the global-best solution more often in later period of the search, the search is guided toward the neighborhood of the best known solution.

Enforcing Pheromone limits

In the MMAS, pheromone values are regulated to stay within the dynamic interval $[\tau_{min,}\tau_{max}]$. Each time a new global best solution S_{bs}(with cost C_{bs}) is found, the value of τ_{max} and τ_{min} is updated as follows[26]:

$$\tau_{max} = \frac{1}{\rho C_{bs}} \tag{3.9}$$

$$\tau_{min} = \tau_{max}/a \tag{3.10}$$

where $a \geq 1.0$ is an algorithmic parameter. At the end of each iteration, the entries in the pheromone matrix are checked and updated to stay within the dynamic allowed limits according to the following formula:

$$\tau_{ij} = \begin{cases} \tau_{min} & if\ \tau_{ij} < \tau_{min} \\ \tau_{max} & if\ \tau_{ij} > \tau_{max} \\ \tau_{ij} & otherwise \end{cases} \tag{3.11}$$

Since the pheromone values are gradually built up, and edges with high amount of pheromone are more likely to be chosen during solution construction, it is possible that a few edges receiving high amount of pheromone in the beginning of the search will be repeatedly chosen in all subsequent stages. Having an upper limit τ_{max} of pheromone values solves this problem. By setting the value of τ_{max} as a function of the solution cost of the global best solution, it is ensured that the highest preference that any edge would receive during solution construction by an ant would not exceed that received by an edge belonging to the global best solution.

The lower limit of pheromone values, τ_{min} helps avoid stagnation. By putting a lower limit on pheromone values, it is ensured that none of the edges would have such a small pheromone value that it will be hardly ever considered in solution construction.

The value of the algorithmic parameter a controls how low the τ_{min} can be compared to the τ_{max}. For small values of a, the pheromone limits are closer to each other and, as a result the edges have similar chance of being selected during solution construction. As the value of a increases, the gap between τ_{max} and τ_{min} also increases. This leads to a wider variety of chances of different edges being selected during solution construction.

Pheromone Initialization and Reinitialization

Pheromone values are initialized with the estimated upper limit $\tau_0 = \frac{1}{C_{sg}}$ where C_{sg} is the initial greedy solution. The pheromone limits are initialized by setting $\tau_{max} = \tau_0$ and $\tau_{min} = \tau_{max}/a$. The value of τ_0 requires careful consideration because a value too high would make the search stay in exploratory stage for too long until the pheromone values are reduced sufficiently due to pheromone evaporation. A very small value on the other hand, would push the search towards a greedy approach because edges belonging to the solutions generated in early iterations would receive considerably more pheromone, and higher chances of being selected during solution construction in subsequent iterations.

Pheromone values are occasionally reinitialized upon detection of stagnation. Stagnation detection is performed at the end of each iteration by calculating the coefficient of variance(CV)[1] of the average cost of the best solutions found in K most recent iterations since last reinitialization, and then comparing the value of CV with a pre-determined threshold. Here, K can be a fixed number (i.e. 100) or provided as an input to the algorithm. If K iterations have not passed since last reinitialization of pheromone values, stagnation detection is not performed. CV is calculated by the following formula:

$$CV = \frac{\sigma}{\mu} \tag{3.12}$$

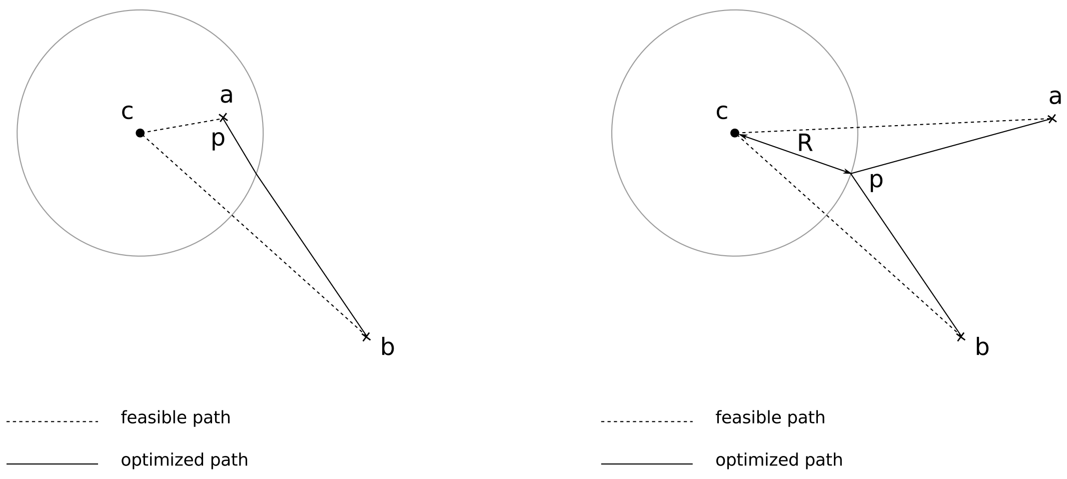
where, μ is the mean costs of best solutions constructed in K most recent iterations, and σ is the standard deviation of those costs. CV is a scale free measurement of the diversity in a set of samples[1]. Its value stays in the interval $[0, 1]$. In this context, CV is a measurement of the diversity of the solutions generated in K most recent iterations. A lower (than the threshold) CV implies that the exploration has narrowed down to a point where the ants are repeatedly generating same or similar solutions. This takes place when there is a significant buildup of pheromone on a subset of edges and very little pheromone on the rest. This causes that subset of edges being frequently used by the ants during solution construction. Upon detection of stagnation, all the entries in the pheromone matrix are set to the current τ_{max} value. This ensures that all the edges will have equal probability of being selected in solution construction of subsequent iterations.

3.4. Refining a feasible trajectory

The feasible trajectory calculated during the first phase using the ACO approach described in section 3.3 is refined in the second phase. The goal here is not to calculate a new trajectory, but to improve the existing one. In order to do so, the notion of the *download-point* is introduced. The download-point of a relay is the point within its region of influence where the MDC arrives in order to download the buffered data from that relay. Since a relay's region of influence is a disk of a given radius, any point on the periphery or inside the disk can serve as the download-point.

If the MDC is already inside the region of influence of the relay, the buffered data can be downloaded without moving any closer to the relay. On the other hand, if the MDC is approaching the relay from outside its region of influence, it is sufficient for the MDC to reach a point on the periphery of the region of influence of the relay in order to download the buffered data. In this case, selecting a point on the periphery of the region of influence reduces the distance traveled by the MDC from its current location to download buffered data from the relay. This concept is explained with an illustration in Figure 3.8.

For an MDC located at the point a, which needs to visit the relay r_c located at the



(a) When either point a (or b) is inside the region of influence.

(b) When both points a and b are outside of the region of influence.

Figure 3.8.: Selecting a download-point on the periphery results in shorter path length.

point c, then move on to point b, the path suggested by the feasible trajectory would be acb. The point c acting as the download-point of the relay located at a potential relay location r_c. However, an optimal point p within the region of influence of the relay can be found such that the resulting path would have shorter length than acb and still allow the MDC to visit the relay. The following cases may arise:

Case-1: The point $a(b)$ is within the region of influence of r_c.

This case is illustrated in 3.8a. The point $a(b)$ serves as the download-point in this case. The MDC can download the buffered data from r_c when it is at $a(b)$. The refined path is a straight line from a to b.

Case-2: When both of the points a, and b are outside of the region of influence of the relay.

This case is illustrated in 3.8b. In this case, a point p within the region of influence can be found such that apb would have a shorter length than acb.

The path refining technique explained above improves the a path by reducing the length; leading to a local optimization of the feasible trajectory.

The proposed solution for refining a given feasible trajectory consists of the following steps:

Step-1: Preprocessing to find out how far the meeting points can be placed from their respective potential relay locations.

Step-2: Finding download-points using either the deterministic heuristic presented in section 3.5 or the $\text{ACO}_{\mathbb{R}}$ approach presented in section 3.6. The list of download-points defines the final trajectory of the MDC; it is referred to as the *final-trajectory* henceforth.

Step-3: Post-processing to find the actual locations of the relay nodes from the suggested download-points and potential relay locations in the feasible trajectory.

The pre-processing and post-processing steps are described next.

3.4.1. Pre-processing for feasible trajectory Refinement

The goal of the pre-processing step is to find, for each potential relay location, the upper limit on the distance of the download-point from the location. A vector denoted by *range* holds this value. The value $range(i)$ denotes the maximum allowable distance of the download point from the ith potential relay location.

As described in section 3.3, some of the locations listed in the given feasible trajectory are $class - I$ locations, while the rest are $class - II$ locations. For all $class - I$ locations, the corresponding *range* entry is set to R.

In the case of $class - II$ relays, they serve only one sensor which is also served by that relay only. However, coverage of that sensor by this relay can be maintained by placing the relay anywhere within the periphery of the region of influence. In an extreme case, the relay could be placed on the periphery of the region of influence of the sensor and the download point would be on the periphery of the region of influence of that relay. In such a situation, the MDC would be communicating with the relay from a point at distance $R + r$ from the sensor. Therefore, the download-point for a relay placed at a $class - II$ location can actually be anywhere within a circle of radius $R + r$ from the sensor served by this relay. Following this rationale, the allowed distance of the download point of all $class-II$ locations are set to $R+r$.

The *range* values for different locations in the previously calculated feasible trajectory of the example scenario are illustrated in Figure 3.9. Potential relay locations not included in the feasible trajectory are omitted to avoid cluttering in the diagram.

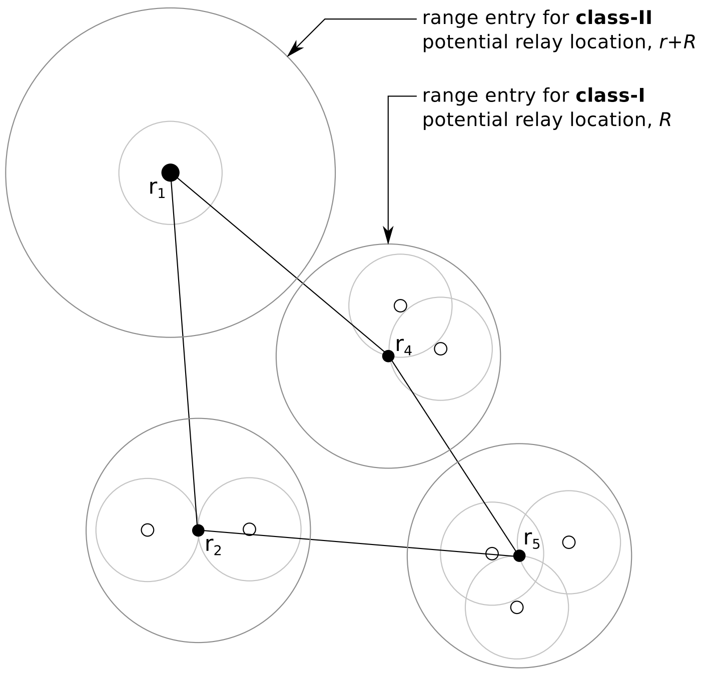

range entry for **class-II** potential relay location, *r+R*
range entry for **class-I** potential relay location, *R*
r_1
r_4
r_2
r_5

Figure 3.9.: The example scenario after setting the range entries for the locations in the feasible trajectory.

3.4.2. Post-processing for relay placement

The goal of the post-processing step is to calculate the final locations of the relays using the feasible-trajectory, and the final-trajectory. A relay is placed in each $class - I$ location of the feasible trajectory. In case of the $class - II$ locations however, the placement of a relay is determined by the corresponding download-point in the final-trajectory.

For each $class - II$ location in the feasible trajectory, a relay is placed at the intersection point of the circle having radius r centered at the relay location and the line segment connecting the corresponding download-point in the final-trajectory. This procedure is illustrated in Figure 3.10. In the given example, c is a class-II potential relay location, the download point for this location is p. A relay node is placed at t, which is the point of intersection between the circle of radius r, centred at c and the line segment cp.

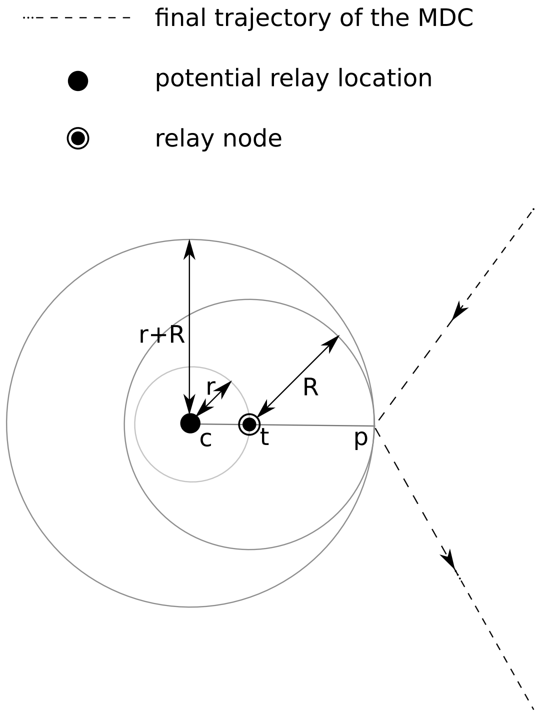


Figure 3.10.: Placement of a relay node for a class-II potential relay location.

3.5. Deterministic Heuristic for Optimizing a Feasible Trajectory

A deterministic heuristic for refining a given feasible trajectory is presented in this section. First, a path refinement operation based on the rationale presented in section 3.4 is defined. This operation is then used to design a deterministic algorithm for performing the actual task of trajectory refinement.

3.5.1. The Path Refinement Operation

The path refinement technique presented in section 3.4 can be seen as an operation performed on a given trajectory. Let T be a feasible trajectory and M be the list of download points of the potential relay locations in T. For a given potential relay location c in the trajectory T, this operation updates the corresponding download point m in M such that the length of the resulting trajectory represented by M is reduced. Let $a = prev(c, M)$ and $b = next(c, M)$. This operation reduces the length of the path from a, through the region of influence of c, to b. It is achieved by transferring the previous download-point m of c to a point p on within the region of influence such that the path apb has shorter length than the path amb.

Based on the relative position of the end points a,b, and c, several cases may arise. The cases are described along with the chosen download-point in each case:

Case-1: Point a (or b) lies inside the circle centred at c having radius $range(c)$:

The point a (or b) is set as the download-point p in this case. Illustrated in Figure 3.11.

Case-2: Perpendicular projection of the point c lies inside the line-

segment ab :

Let p' be the projection of the point c on the line-segment ab. Illustrated in Figure 3.12. Then, the following two sub cases may arise:

Case-2a: The point p' is inside the circle centred at c having radius $range(c)$:

The projection point p' is set as the download-point p.

Case-2b: The point p' is outside the circle centred at c having radius $range(c)$:

The intersection point between the angle-bisector of $\angle acb$ and the circle centred at c, having radius $range(c)$ is set as the download-point, p.

Case-3: Perpendicular projection of the point c lies outside the line-segment ab :

This case is illustrated in Figure 3.13. The following two sub cases may arise:

Case-3a: The point c is closer to point a, than b :

The intersection point of the line segment ac and the periphery of the circle centred at c having radius $range(c)$ is set as the download-point p.

Case-3b: The point c is closer to point b, than a :

The intersection point of the line segment bc and the periphery of the circle centred at c having radius $range(c)$ is set as the download-point p.

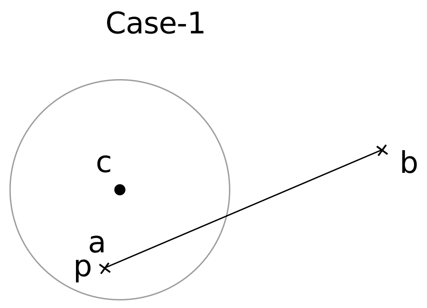


Figure 3.11.: Case-1 of finding download point by deterministic heuristic.

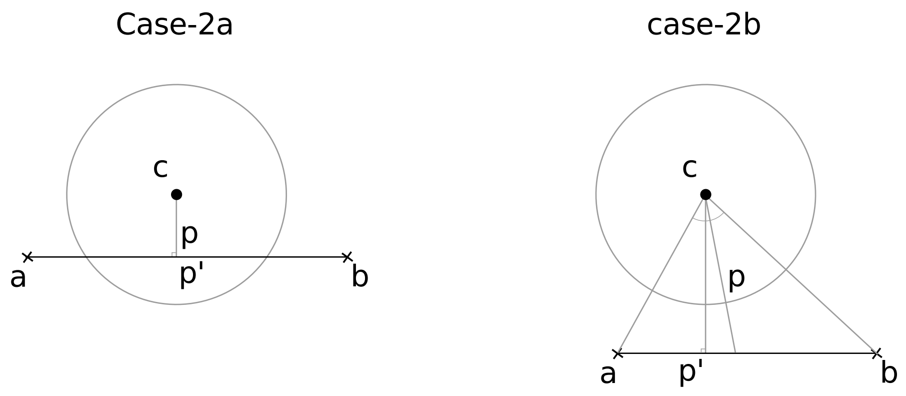


Figure 3.12.: Case-2 of finding download point by deterministic heuristic.

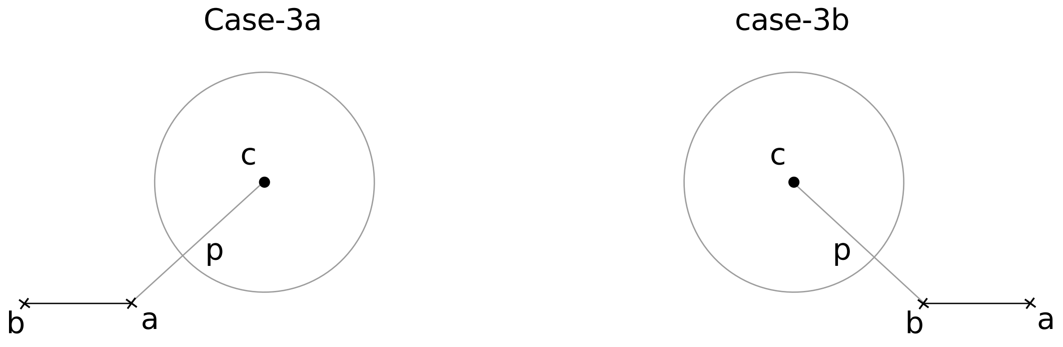


Figure 3.13.: Case-3 of finding download point by deterministic heuristic.

3.5.2. A Deterministic Algorithm for Optimizing a Feasible Trajectory

An algorithm for optimizing a feasible trajectory by representative application of the path refinement operation is presented in Algorithm Algorithm 3.5. Taking a trajectory T, the vector *range*, and a real number *threshold*, the algorithm delivers M, an improved version of the input trajectory T.

Algorithm 3.5 DeterministicTrajectoryOptimization

```
Input: $T$
       $range$
       $threshold$
Output: $M$ = an optimized trajectory expressed as a list of points
 1: $M \leftarrow$copy of $T$
 2: $C_{previous} \leftarrow \infty$
 3: $C_{current} \leftarrow cost(T)$
 4: $\Delta_{cost} \leftarrow \infty$
 5: $c \leftarrow$the first entry in $T$
 6: while $\Delta_{cost} > threshold$ do
 7:     $a \leftarrow index(previous(c), M)$
 8:     $b \leftarrow index(next(c), M)$
 9:     $M(c) \leftarrow$PathRefinement($M(a)$,$M(b)$,$c$,$range(c)$)
10:     $c \leftarrow next(c)$
11:     $C_{current} \leftarrow cost(M)$
12:     $\Delta_{cost} \leftarrow c_{current} - c_{previous}$
13:     $c_{previos} \leftarrow c_{current}$
```

The sub-procedure $PathRefinement$ performs according to the steps described in subsection 3.5.1.

The algorithm starts by making an exact copy M of the input trajectory T. This is illustrated in Figure 3.14. The points r_1, r_2, r_4 and r_5 denote potential relay locations that were selected in the first phase. The points p_1, p_2, p_4 and p_5 are the download-points being calculated. At this stage, the feasible trajectory T and M are the same.

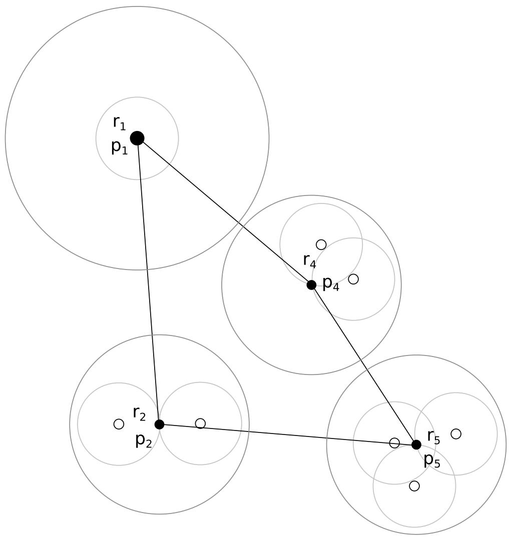


Figure 3.14.: Initializing M as a copy of T in Algorithm Algorithm 3.5.

A few variables are initialized to keep track of the current cost and previous cost of the trajectory and the cost difference. The first entry in T is set to the current location c. After that, the following actions are performed repeatedly until the cost difference becomes smaller than *threshold*.

Let $a(b)$ the index of the preceding(succeeding) entry of c in T. The path $M(a)-c-M(b)$ is refined using the path refinement operation presented in subsection 3.5.1. Then, c is set to the next entry in T and the costs are updated.

The step by step execution of this procedure on the example scenario is shown in Figure 3.15 through Figure 3.18, with the final result shown in Figure 3.18.

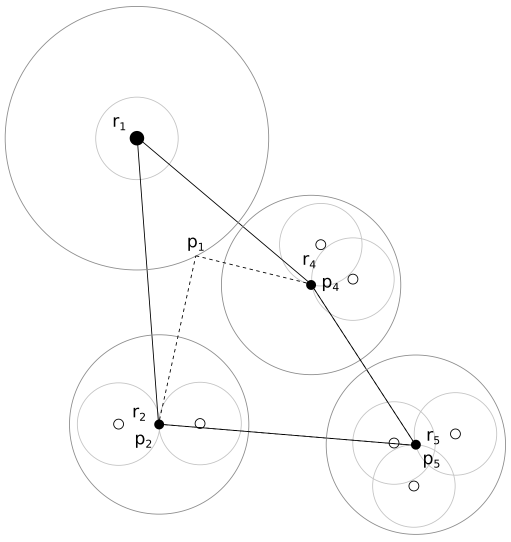


Figure 3.15.: Refining of the path $p_4 - r_1 - p_2$, $p_1$is the new download-point.

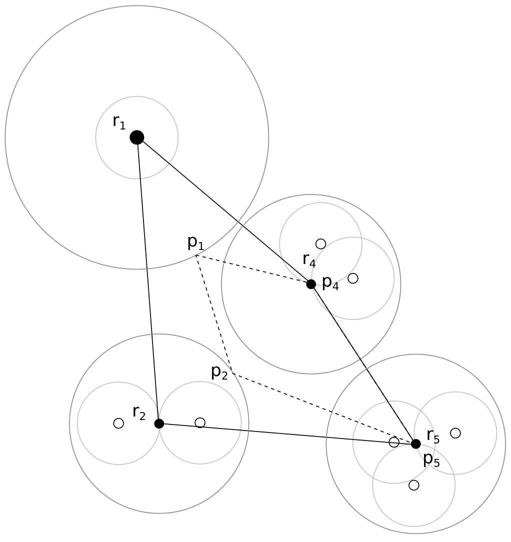


Figure 3.16.: Refining of the path $p_1 - r_2 - p_5$, $p_2$is the new download-point.

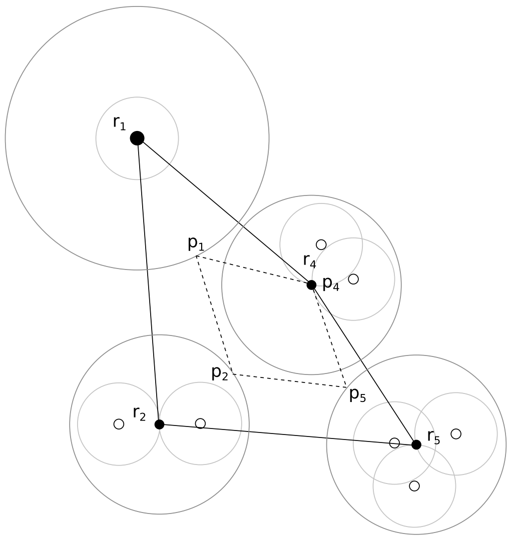


Figure 3.17.: Refining of the path $p_2 - r_5 - p_4$, $p_5$is the new download-point.

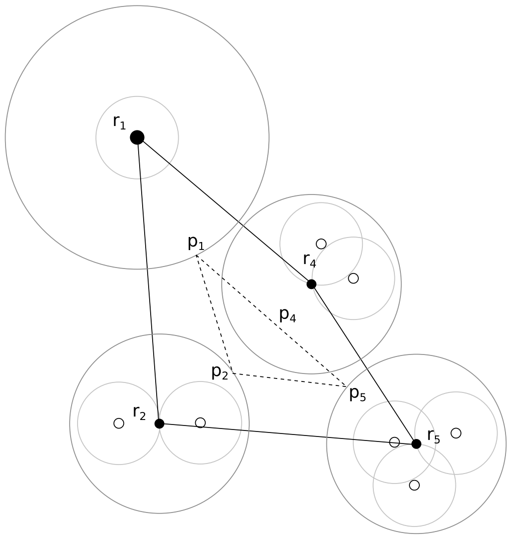


Figure 3.18.: Refining of the path $p_2 - r_5 - p_4$, $p_4$is the new download-point.

3.6. ACO$_\mathbb{R}$ Approach for Optimizing the Feasible Trajectory

A Continuous Ant Colony Optimization (ACO$_\mathbb{R}$) approach for refining a feasible trajectory is presented in this section. Given a feasible trajectory T, the goal of the proposed optimization approach is to find an improved trajectory M, consisting of the download points of the potential relay locations in T. It is assumed that the download point of a given location lies on the periphery of the region of influence. Such a download point can be expressed by the angle with respect to an arbitrary axis, as illustrated in Figure 3.19. For a potential relay location c, the download point p at a distance r can be specified by the angle a. This angle is called the *hitting angle*; generally the hitting angle lies inside the interval $[-\pi, \pi)$. This concept is adapted from [88].

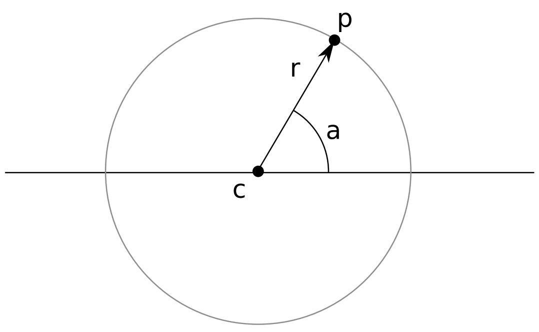


Figure 3.19.: Hitting angle.

For two given relay locations a and b in T, such that $c_a = prev(c_b, T)$ (in other words, c_b is visited right after c_a), let A and B denote two circles representing their respective areas of influence. For the circles A and B centred at points c_a, c_b and having radius r_a, r_b respectively, all possible download points in B, when visited after A, can be represented as a result of the intersection between the periphery of B and the lines passing through the points in A. This is illustrated in Figure 3.20.

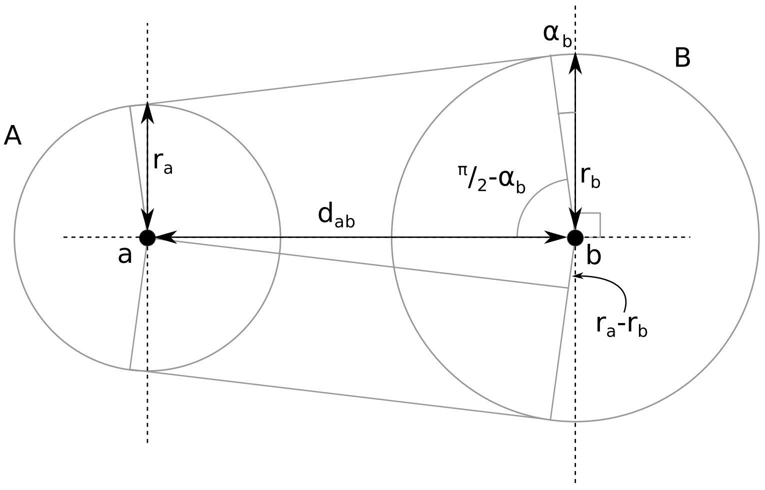


Figure 3.20.: Distribution of hitting angle.

Let d_{ab} denote the distance between a and b. By choosing the line segment connecting a and b as the axis for expressing the hitting angle, the distribution of the hitting angles can be bounded within the interval $[-\frac{\pi}{2} + \alpha_b, \frac{\pi}{2} - \alpha_b]$ where α_b is measured by:

$$\alpha_b = arcsin\left(\frac{|r_a - r_b|}{d_{ab}}\right) \tag{3.13}$$

Following this rationale, the line segment connecting a with b is chosen as the axis from which the hitting angle is measured. The angle between this line segment and the X-axis is called the *offset angle,* denoted by ω_b, of the potential relay location b. This is illustrated in Figure 3.21. The coordinates of the corresponding download point p_b can be found by the formula:

$$p_x = b_x + range(b) \times cos(\omega_b + \theta_b) \tag{3.14}$$

$$p_y = b_y + range(b) \times sin(\omega_b + \theta_b) \tag{3.15}$$

where, p_x (p_y) denotes the $X(Y)$ coordinate of the point p, and b_x (b_y) denotes the $X(Y)$ coordinate of the point b.

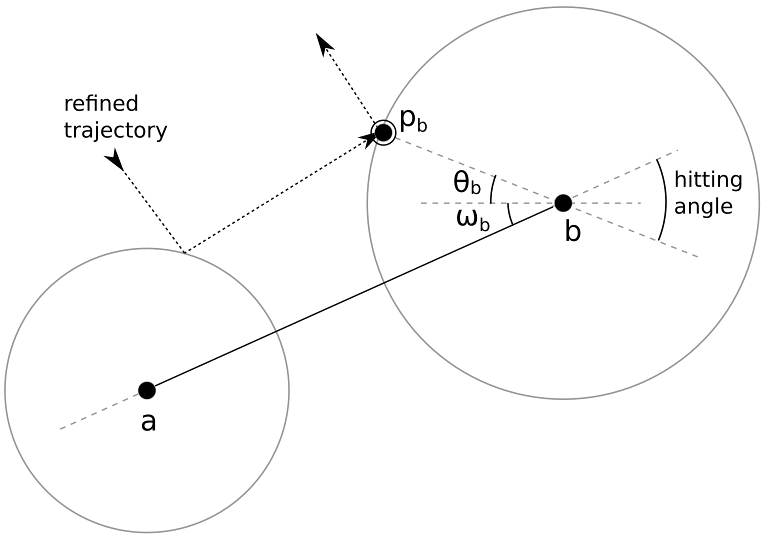


Figure 3.21.: Offset angle.

Since the order of visiting the potential relay locations is already defined by the feasible trajectory, it is sufficient to calculate the hitting angles of each of the locations in the feasible trajectory, such that the cost of the resulting trajectory is optimized. Since the hitting angles are real numbers, the problem boils down to finding N real numbers, each denoting the hitting angle for a location in the feasible trajectory. This multivalued real optimization problem is solved by the proposed $\text{ACO}_{\mathbb{R}}$ approach. A solution instance S is thus expressed by a vector containing N real numbers (Table 3.1), the value $S(i)$ denotes θ_i, the hitting angle of the potential relay location i in T.

	1	2	3	$\cdots$	N
S	θ_1	θ_2	θ_3	$\cdots$	θ_N

Table 3.1.: A solution instance of the proposed $\text{ACO}_{\mathbb{R}}$ heuristic.

In order to estimate the cost of such a solution instance, it is converted to a trajectory, expressed as a list of points in the euclidean space, using the Algorithm 3.6.

Here M is the output produced by Algorithm 3.6.

Algorithm 3.6 TrajectoryFromHittingAngles.

```
Input: $T$, feasible trajectory
    $S$, a vector holding the angles $\theta_1$, $\theta_2$, ... ,$\theta_N$
    $\omega_1$, $\omega_2$, ... ,$\omega_N$, the offset angles
Output: $M$, a refined trajectory
1: $M \leftarrow T$
2: for all $r \in T$ do
3:     $\theta_r \leftarrow S(r)$
4:     $x \leftarrow r_x + range(r) \times cos(\theta_r + \omega_r)$
5:     $y \leftarrow r_y + range(r) \times sin(\theta_r + \omega_r)$
6:     $t \leftarrow Point(x, y)$
7:     $M(r) \leftarrow t$
```

3.6.1. Proposed ACO$_{\mathbb{R}}$ heuristic

The proposed ACO$_{\mathbb{R}}$ heuristic utilizes the concept of hitting angles to refine a given feasible trajectory. The algorithm begins with initializing the pheromone table. After initializing the pheromone table, the iterative part of the algorithm begins. The following actions are performed during each iteration until a stopping criteria is satisfied:

- Solution Construction:

  A number of ants are employed to build tentative solutions by consulting the existing entries in a pheromone table. Here, m denotes the number of ants employed, and is an algorithmic parameter provided as an input.

- Pheromone Update:

  The pheromone table is updated by replacing poor quality solutions in the table with better solutions constructed by the ants in this iteration.

The different components of the proposes ACO$_{\mathbb{R}}$ heuristic is described in the remainder of this section.

Pheromone Representation and Initialization

As described in section 2.7, the pheromone in $ACO_{\mathbb{R}}$ is represented by a pheromone table consisting of a number of entries; each entry denotes a solution instance (Table 2.2). Here, k is an algorithmic parameter provided as an input, and represents the total number of entries in the pheromone table. Results of experimenting with different values of k is presented in Chapter 4. The jth column in the table correspond to the jth potential relay location in the feasible trajectory and holds the value of θ_j for a particular solution instance.

To begin with, the pheromone table is initialized by inserting a greedy solution S_g containing $\theta_j = 0$, for each potential relay location j in T. This solution results in a trajectory, where the download point for each potential relay location in T is the point where the line segment connecting this location with its predecessor intersects with the periphery of the region of influence of the relay. The trajectory resulting from this initial greedy solution for the example scenario is illustrated in Figure 3.22.

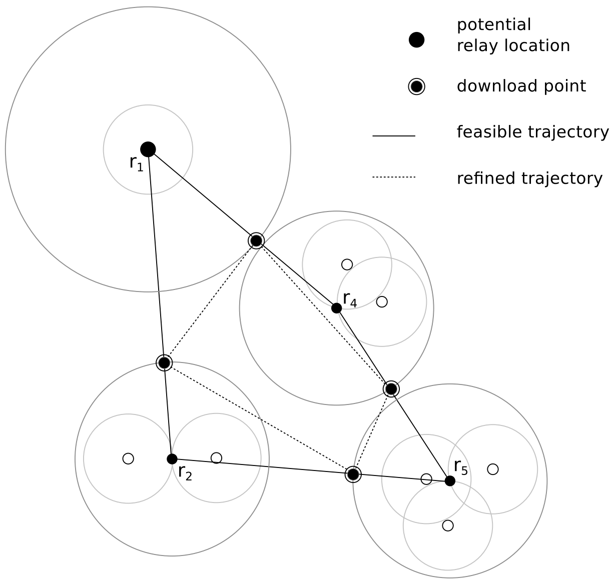


Figure 3.22.: Trajectory resulting from initial greedy solution of $ACO_{\mathbb{R}}$.

Following the initialization of the pheromone table, solution construction and pheromone update is performed iteratively, until a stopping criteria is met. Solution constructions and pheromone updating have been described earlier in section 2.7. Occasionally the pheromone values are reinitialized upon detection of stagnation. These components of the algorithm are described next. The entire algorithm is summarized in Algorithm 3.7.

Algorithm 3.7 $ACO_{\mathbb{R}}$ForRefiningFeasibleTrajectory

```
Input: $T$, feasible trajectory
       $I$, number of iterations
       $t$, threshold
Output: $s$, an $ACO_{\mathbb{R}}$ solution instance
 1: $S_g \leftarrow \emptyset$
 2: for all $r \in T$ do
 3:    $S_g(r) \leftarrow 0$
 4: $C_g \leftarrow cost(S_g)$
 5: $\mathcal{T} \leftarrow \emptyset$
 6: insert($\mathcal{T}$,$S_g$)
 7: $i \leftarrow 1$
 8: $CV \leftarrow 1$
 9: while $i \neq I$or $CV > t$ do
10:    $G \leftarrow \emptyset$
11:    $W \leftarrow \emptyset$
12:    for all $m \in [1, m]$ do
13:       $S \leftarrow \emptyset$
14:       for all $j \in T$ do
15:          $S(j) \leftarrow$sample($\mathcal{T}, j$)
16:    $G(m) \leftarrow S$
17:    $W(m) \leftarrow cost(S)$
18:    for all $s \in W$ do
19:       insert($\mathcal{T}$,$s$)
20:    sort($\mathcal{T}$)
21:    for all $p \in [k+1, k+m]$ do
22:       remove($\mathcal{T}(p)$)
23:    $CV \leftarrow \frac{sd(W)}{mean(W)}$
24:    $i \leftarrow i + 1$
```

Stagnation Detection and Pheromone Reinitialization

Stagnation detection is performed during the step by step construction of a solution. Using the procedure described in section 2.7, the jth step of the solution construction consists of taking a random sample, based on the values in the jth column of the pheromone table. The stagnation detection is performed by comparing the coefficient of variation (CV) of the values in the jth column with a threshold. Since the CV in any sample set provides a scale-free measure of the variability of the samples[1], a lower than threshold value of CV in this case indicates that all the random samples for hitting angles are being drawn from a relatively narrow range. This phenomena is interpreted as an indication of stagnation. Upon detecting stagnation, the stagnant column of the pheromone table is reinitialized by setting all the entries in that column to 0.

Pheromone Update and Stopping Criteria

The cost of the solutions constructed during an iteration is calculated at the end of an iteration. These k new solutions are inserted into the pheromone table; the table now contains $k+m$ solution. These are sorted by their corresponding weights, calculated using Equation 2.11. The best k solution are kept and the rest are discarded.

The stopping criteria is adapted from [26]. The CV of the costs of the solutions constructed during an iteration is calculated for detecting convergence of the search procedure. When the costs of different solutions constructed by the ants during an iteration fall within a narrow range, that phenomena is interpreted as convergence of the search procedure. The CV of cost values is compared with a previously determined threshold value. The searching is stopped when the CV is lower than the threshold or a certain number of iterations have been performed, whichever occurs first.

4. Experimental Results

The results of the experiments performed using the proposed approach to jointly solving the Relay Node Placement and Trajectory calculation (RNPT) problem are presented in this chapter. Three variations of ACO i.e., the Ant System(AS), the Elitist Ant System(EAS), and the Max Min Ant System(MMAS) were discussed in Chapter 2. A comparative study was carried out in order to find out the most suitable approach for solving our problem. The most suitable approach based on the initial experimental results, was chosen for further experimentation. The effect of varying different algorithmic parameters and the choice of different algorithmic components were studied next. In the last section, a comparison between the deterministic approach, and the $\text{ACO}_{\mathbb{R}}$ based approach for trajectory refinement is presented. The following performance metrics, which are relevant for a particular experiment, were used to compare the outputs of the experiments performed:

- Feasible trajectory length
- Refined trajectory length
- Number of relays used
- Time per iteration

For our simulations, we focused on two network sizes $n = 25$, and $n = 50$, where n denotes the number of sensors, in sensing fields of dimension 150×150 and 200×200

respectively. For each network size, 10 different sensor distributions were randomly generated. For the networks with $n = 25$, 500 iterations were performed during each run, while 1000 were performed for networks with $n = 50$. The results of the initial greedy approach on the test networks are listed in Table A.1, and Table A.2.

Two different cost metrics were discussed in section 3.3:

- The metric $C^1_s = l_s \sum_{u \in s} d_{u,next(u)}$ estimates the cost of a solution s as a product of the number of relays used and the trajectory length.
- The metric $C^2_s = \sum_{u \in s} d_{u,next(u)}$ estimates the cost as the total trajectory length, ignoring the contribution of the number of relays used.

For each set of network parameters, experiments were performed using different variants of ACO approaches using either of the two cost metrics. The experiments and their results are presented next.

4.1. Experiments on ACO Variants Using C^1

The three ACO variants were compared by running a series of experiments. The three main algorithmic parameters α, β, and ρ were set to suggested standard values, listed in Table 4.1, for solving the TSP problem[26].

ACO Approach	α	β	ρ
AS	1.0	2.0	0.50
EAS	1.0	2.0	0.50
MMAS	1.0	2.0	0.02

Table 4.1.: Suggested[26] algorithmic parameters for solving TSP.

The cost metric C^1 was used during these experiments. The detailed results are listed in Table A.3, Table A.4 and are illustrated in Figure 4.1, and Figure 4.2 respectively. The presented values are the averages of 5 runs on each network.

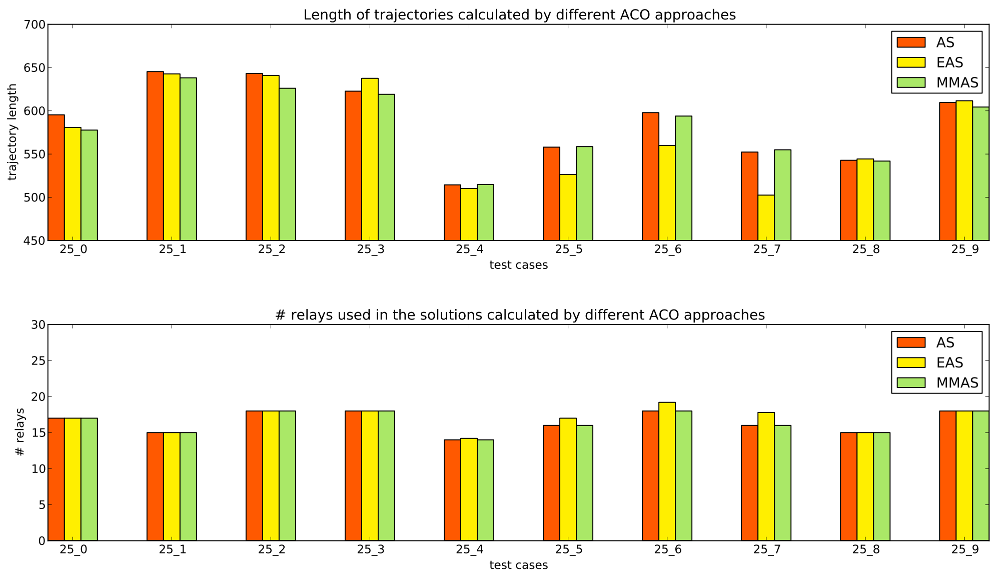


Figure 4.1.: Results of applying different ACO approaches on networks with 25 sensors.

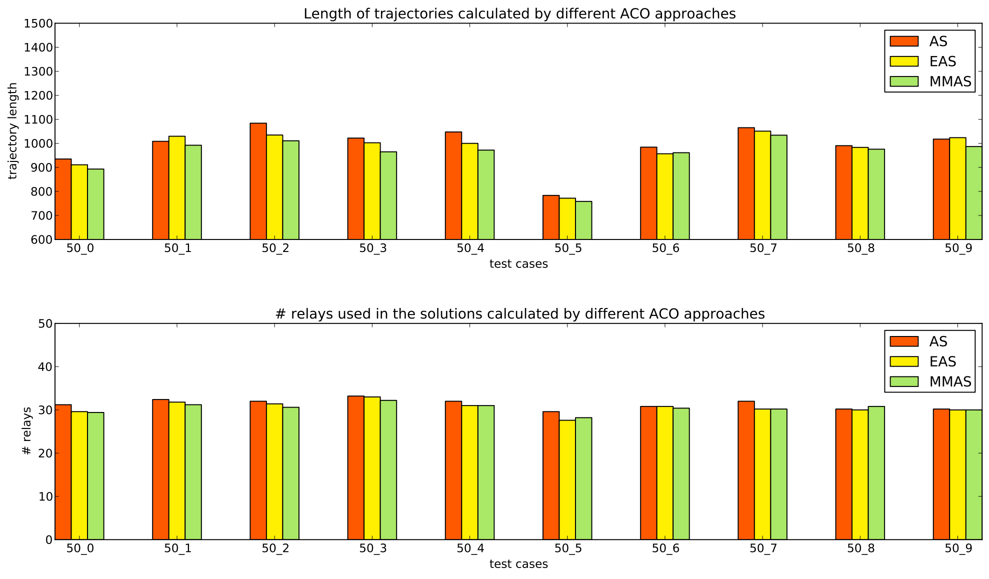


Figure 4.2.: Results of applying different ACO approaches on networks with 50 sensors.

None of the three ACO approaches performed consistently better than the others for the networks with $n = 25$. For the networks with $n = 50$ however, AS performed the worst, and MMAS performed the best for most cases. The results of EAS was found to be comparable to that of MMAS, although a little inferior.

However, it was noted during this experiment that the running time for MMAS is longer, compared to that of EAS for the same input. In order to gain a comparative understanding of the running times of the two approaches, the average running times per iteration for each network was measured. The results are presented in Figure 4.3. The average running time per iteration for EAS was shorter than that of MMAS for all the networks except one. The difference in running time per iteration was significantly larger for networks with $n = 50$, when compared to that of the ones with $n = 25$. Due to the inferior performance, AS was not considered for the next set of experiments.

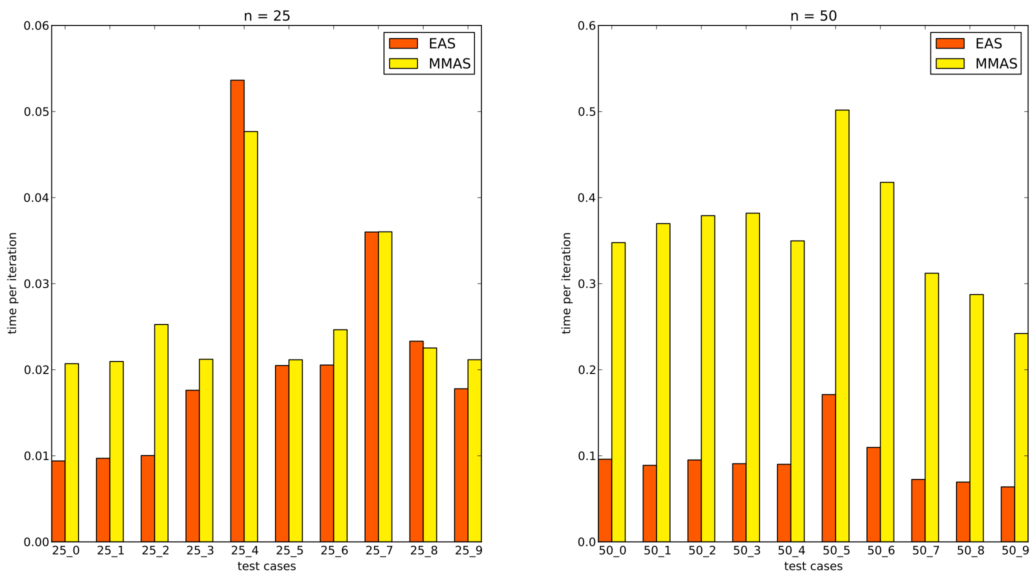
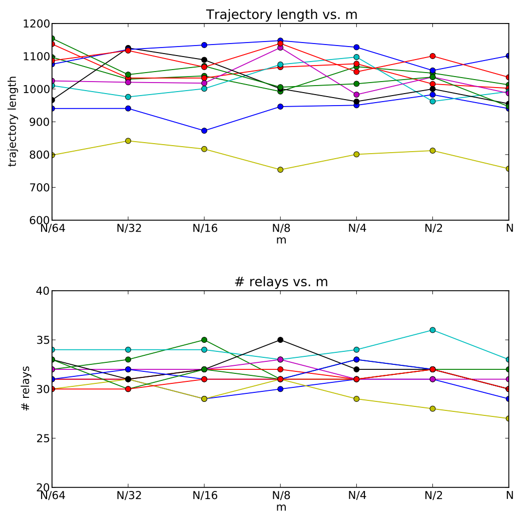
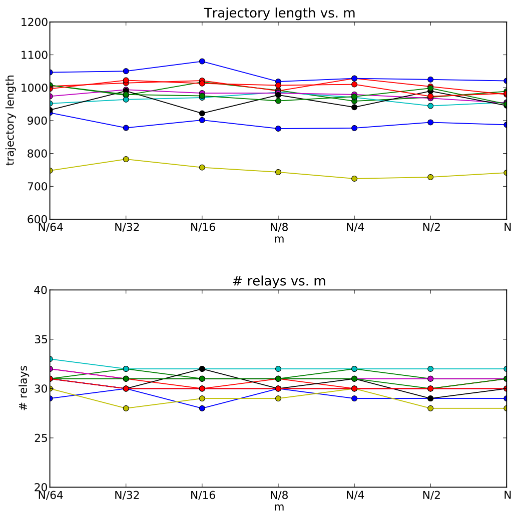


Figure 4.3.: Comparison of time per iteration for EAS, and MMAS for different networks.



(a) EAS, on 10 networks with 50 sensors each.



(b) MMAS, on 10 networks with 50 sensors each

Figure 4.4.: Trajectory length, and # relays used VS. # of ants employed.

4.1.1. Number of Ants Employed

The running time increases with m the number of ants employed in each iteration. In the case of TSP, using the same number of ants as the number of cities has been suggested[26]. For our problem however, setting $m = N$, where N denotes the number of potential relay locations, results in prohibitively large running time. In order to gain an understanding of the affect of varying m on the quality of the solution quality, experiments were performed by running the EAS, and MMAS approach on the test networks with $n = 50$, by varying the value of m within the set $\{N, \frac{N}{2}, \frac{N}{4}, \frac{N}{8}, \frac{N}{16}, \frac{N}{32}, \frac{N}{64}\}$. The results are shown in Figure 4.4. To measure the degree of interdependence between the two main performance metrics and m, the Pearson's correlation coefficients[67] were calculated. The results are shown in Table 4.2. Stronger negative correlation in the case of EAS shows that the result improves as m increases. In the case of MMAS on the other hand, the value of m has less measurable influence on the result. It is also evident from the plot in

Figure 4.4 that, the result of MMAS varies very little with m.

Performance metric	EAS	MMAS
cor(# relays, m)	-0.274	-0.155
cor(Trajectory length, m)	-0.183	-0.095

Table 4.2.: Correlation between m and performance metrics when cost metric C^1 is used.

4.2. Experiments on ACO Variants Using C^2

Only the length of the trajectory is considered in the cost metrics C^2. This metric was suggested for solving TSP[26]. The same set of experiments as the ones presented in the previous section were performed on the three ACO variants using cost metric C^2. The detailed results are listed in Table A.5, Table A.6 and presented in Figure 4.5, Figure 4.6, and Figure 4.7. In case of using C^2 as the cost metric, EAS was found to perform better than AS and MMAS both in terms of trajectory length, and number of potential relay locations. AS was again found to perform the worst for all the networks. Due to this inferior performance, AS was not considered for any further experimentations. When the results of EAS, and MMAS were compared by varying m, MMAS was found to consistently produce good results while EAS produced result that varied with m. The Pearson's correlations between the two main performance metrics and m are shown in Table 4.2. Stronger negative correlation in the case of EAS shows that the result improves as m increases. In the case of MMAS on the other hand, the value of m has less measurable influence on the result.

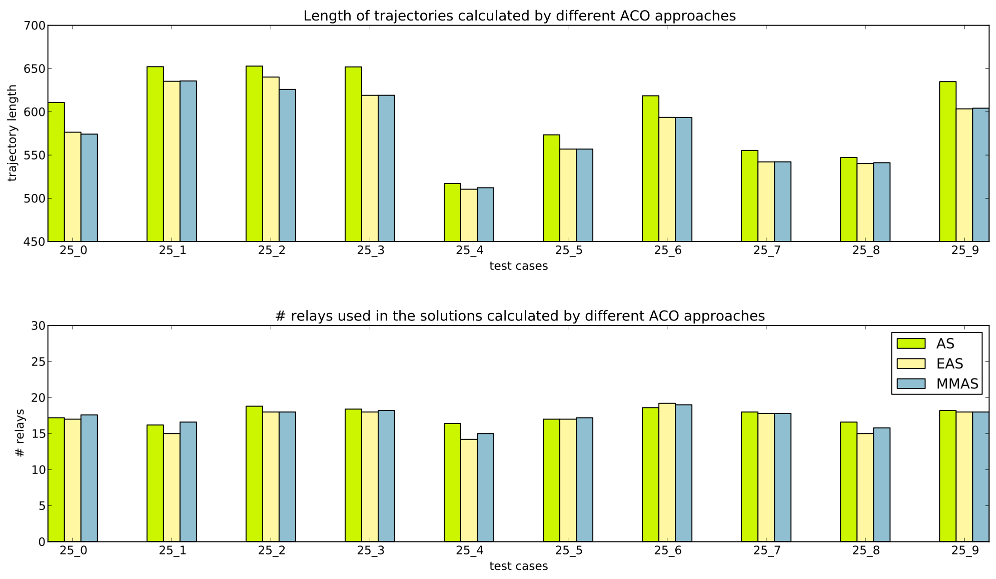


Figure 4.5.: Results of applying different ACO approaches on networks with 25 sensors, using cost metric C^2

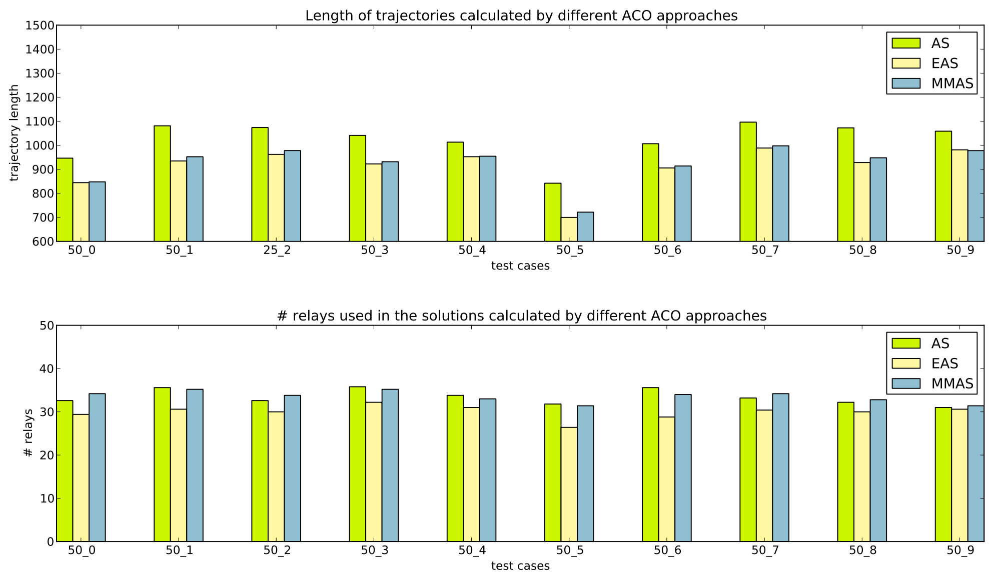
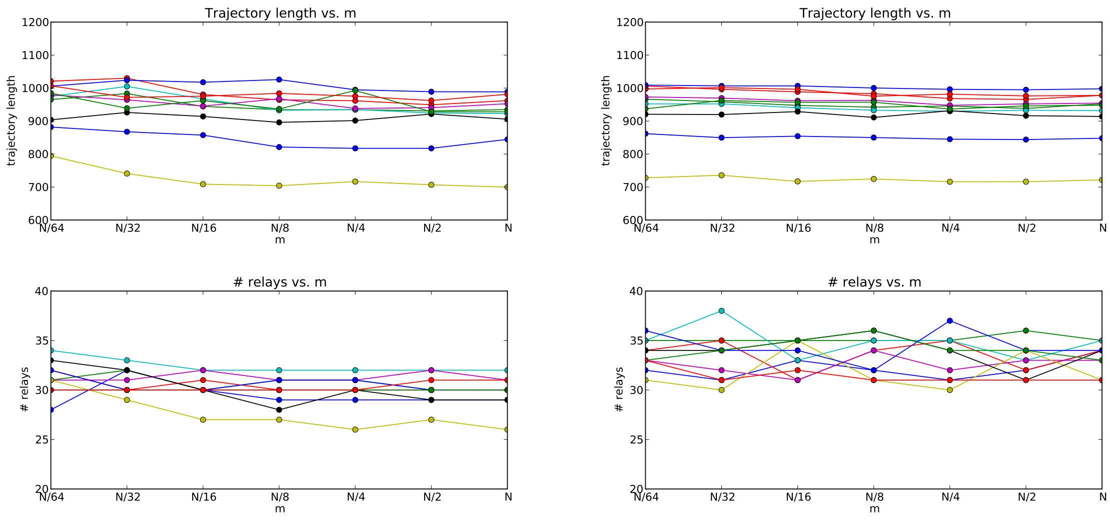


Figure 4.6.: Results of applying different ACO approaches on networks with 50 sensors, using cost metric C^2



(a) EAS, on 10 networks with 50 sensors each.

(b) MMAS, on 10 networks with 50 sensors each.

Figure 4.7.: Trajectory length, and # relays used VS. # of ants employed, using cost metric C^2

Performance metric	EAS	MMAS
cor(# relays, m)	-0.159	-0.074
cor(Trajectory length, m)	-0.204	-0.092

Table 4.3.: Correlation between m and performance metrics when cost metric C^2 is used.

4.3. Comparison of EAS, MMAS Using C^1, and C^2

Side by side comparison of the results produced by the greedy algorithm(section 3.3), EAS, and MMAS using different cost metrics for networks with $n = 25$, and $n = 50$ are presented in Figure 4.8, and Figure 4.9 respectively. For most networks with $n = 25$, the ACO approaches perform better than the greedy algorithm in terms of the number of relays used and trajectory length. MMAS with C^1 consistently produced the best results in terms of the number of relays. But none of the combinations of ACO algorithm and cost metric produced consistently better result than the others in terms of the trajectory length. For networks with $n = 50$, the ACO approaches

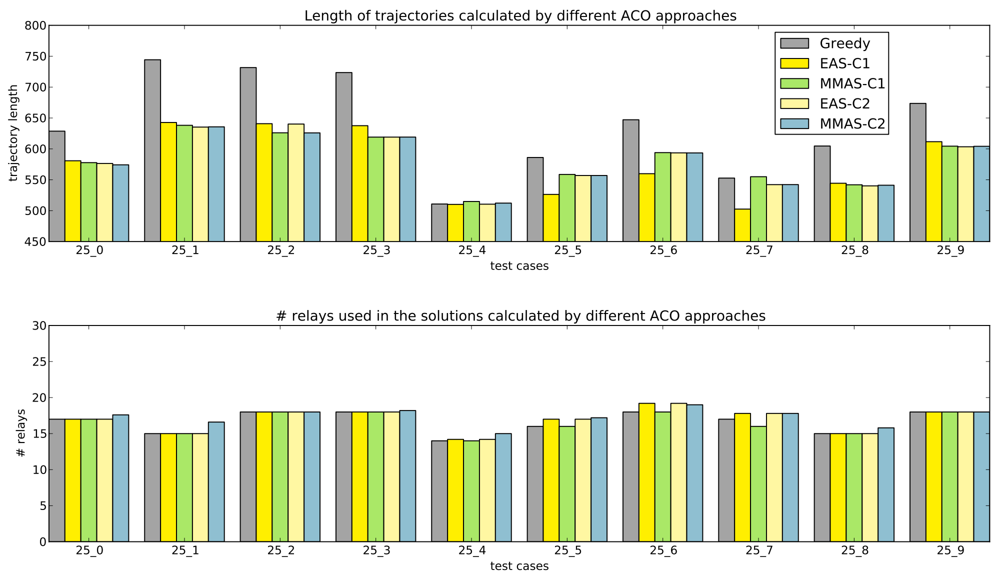


Figure 4.8.: Results of EAS, and MMAS using different cost metrics, on networks with $n = 25$.

perform better than the greedy algorithm in terms of trajectory length. But for most of the networks, the greedy algorithm performs best in terms of the number of realys used. EAS with C^2 consistently produced the best results among ACO approaches in terms of trajectory length. In most cases MMAS with C^1 produced the best results among the ACO approaches in terms of the number of relays used but, the results of EAS with C^1, and C^2 were comparable.

It was concluded based on these observations that, MMAS performs the best among the ACO approaches, but these results are produced at the cost of longer running time. However, EAS provides a balance between running time and solution quality. MMAS with cost metric C^1 was identified as the most suitable ACO approach for jointly solving the RNP and calculation of a trajectory. This combination is used in subsequent experiments. However, EAS with cost metric C^2 is recommended if higher priority is given to the trajectory length, or the constraints are somewhat relaxed to allow more than the minimum number of relays to find a shorter trajectory

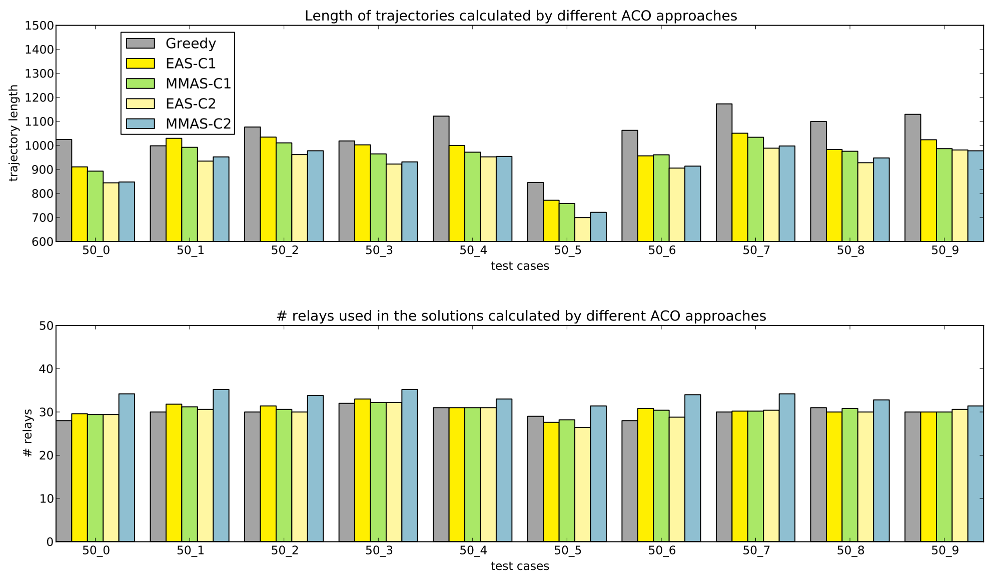


Figure 4.9.: Results of EAS, and MMAS using different cost metrics, on networks with $n = 50$.

length.

4.4. Heuristic information

The heuristic information is a measure of attractiveness of a move during the construction of a solution. Two different versions of heuristic information were mentioned in section 3.3:

- $\eta^1_{ij} = \frac{u(j)}{d_{ij}}$, takes into account the distance between the ith and jth potential relay node location and the number of uncovered sensors that can be covered by selecting the jth potential relay node location.
- $\eta^2_{ij} = \frac{1}{d_{ij}}$, takes into account only the distance between the ith and jth potential relay node location.

The two versions of the heuristic information were compared by running the MMAS using C^1on all the test networks twice; using a different heuristic information each time. The results are shown in Figure 4.10, and Figure 4.11. The results were comparable for networks with $n = 25$. But for networks with $n = 50$, the trajectory lengths were significantly shorter when η^2 was used while the number of relays used stayed comparable. Based on this observation, it was concluded that η^2 was better suited than η^1 for solving the RNPT problem.

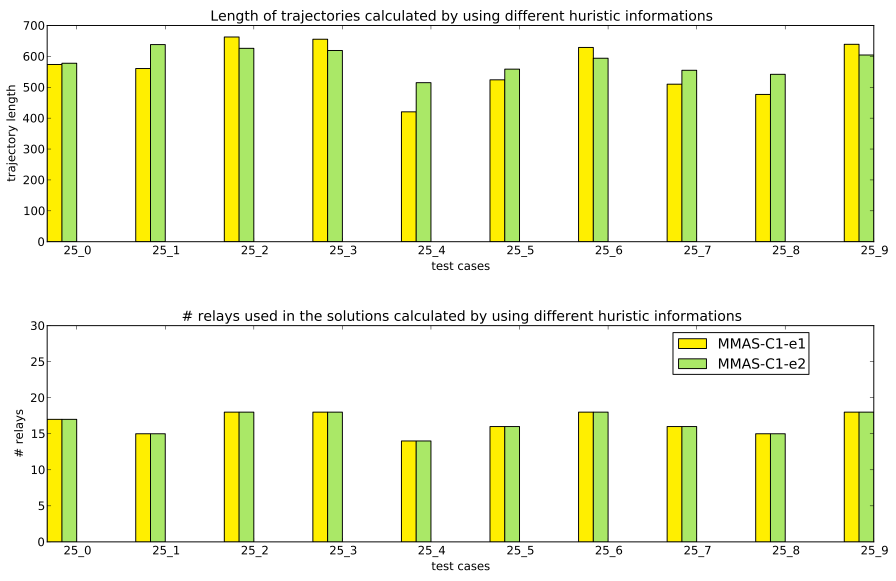


Figure 4.10.: Comparison of different heuristic informations on networks with $n = 25$.

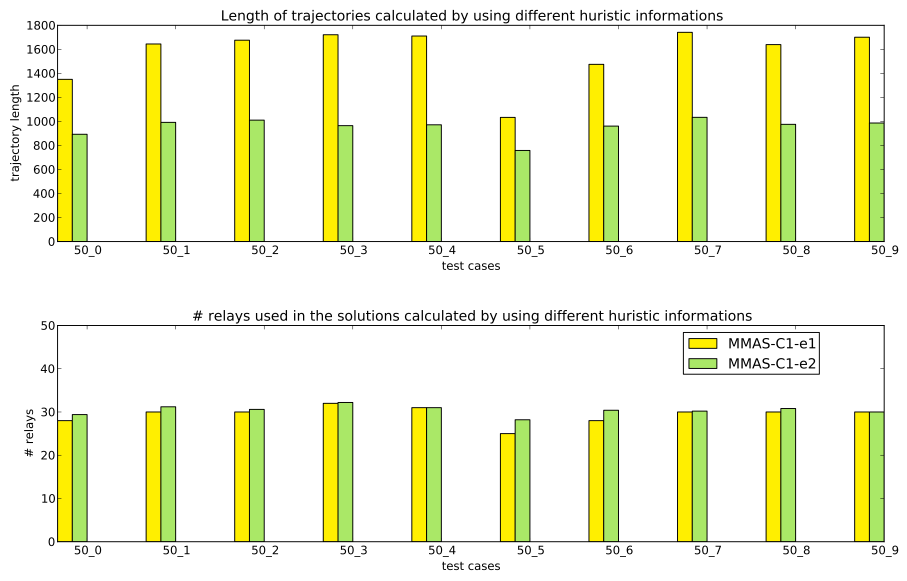


Figure 4.11.: Comparison of different heuristic informations on networks with $n = 50$.

4.5. Strategies for Calculating Potential Relay Node Locations

Three different strategies for calculating potential relay node locations were mentioned in section 3.3. These strategies differ from each other in the handling of disconnected sensors, or sensors whose regions of influence do not overlap with that of any other sensor. The three proposes strategies were:

- selecting the location of the sensor as the potential relay location;
- selecting 4 points on the boundary of the sensor's region of influence, evenly spaced from each other;
- selecting 8 points on the boundary of the sensor's region of influence, evenly spaced from each other.

For sensors whose regions of influence overlap, the potential relay locations are always the intersection points of the boundaries of their respective regions of influence. Experiments were performed to find out the efficacy of the three strategies; using MMAS with cost metric C^1. The results are listed inTable A.7, and Table A.8. The result of first, and second phase of the proposed approach (section 3.3, section 3.4) are listed as feasible trajectory and refined trajectory respectively. The results are also shown in Figure 4.12, and Figure 4.13. For networks with $n = 25$, the shortest feasible trajectories were produced when the 8-point strategy is used, but the refined trajectories produced from different potential relay location strategies were comparable. All strategies produced same number of relays for all the networks with $n = 25$. The average time per iteration was the shortest when the centre point strategy was used, and longest average time per iteration resulted from the 8-point strategy. For networks with $n = 50$, the average feasible trajectory lengths were comparable, but the refined trajectory lengths were significantly reduced when the 8-point strategy was used. The average number of relays used were the lowest when the 8-point strategy is used, with next to best was found when the centre point strategy was used. The average running time per iteration was the shortest in case of the centre point strategy, and the longest in case of the 8-points strategy. Based on these results, it was concluded that the 8-points strategy produces the best results while the centre point strategy provides a balance between result quality and running time.

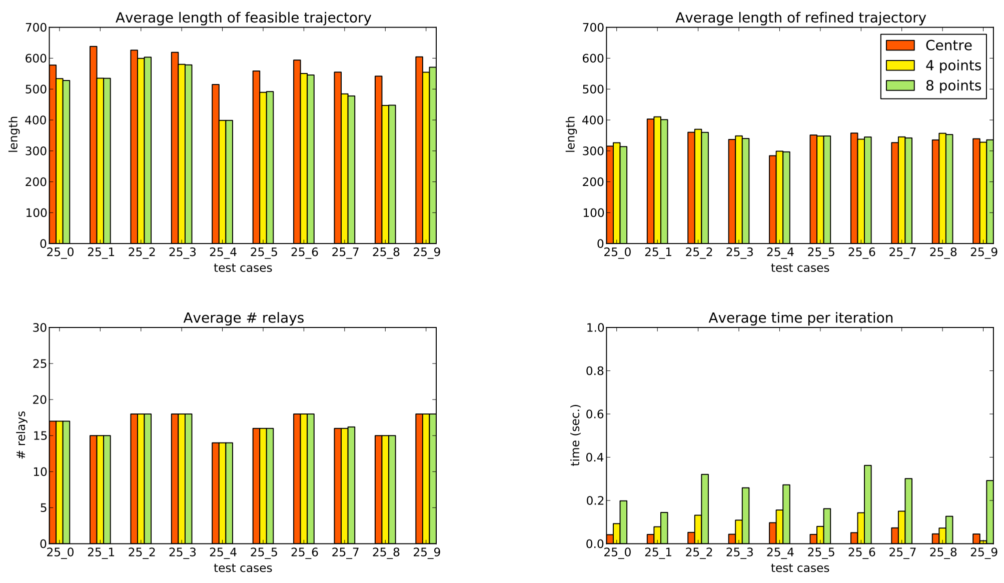


Figure 4.12.: Comparison different strategies for calculating potential relay locations on networks with $n = 25$.

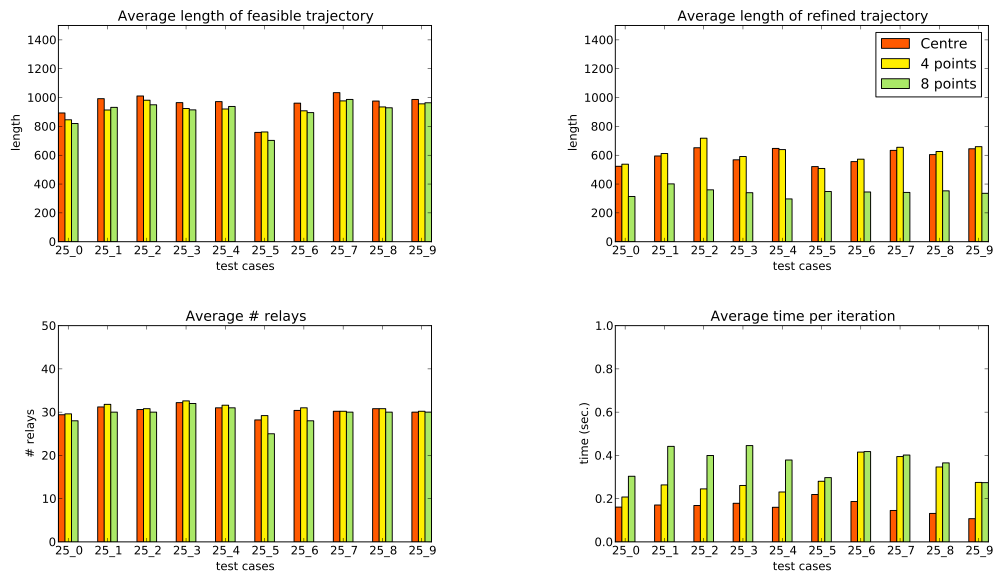
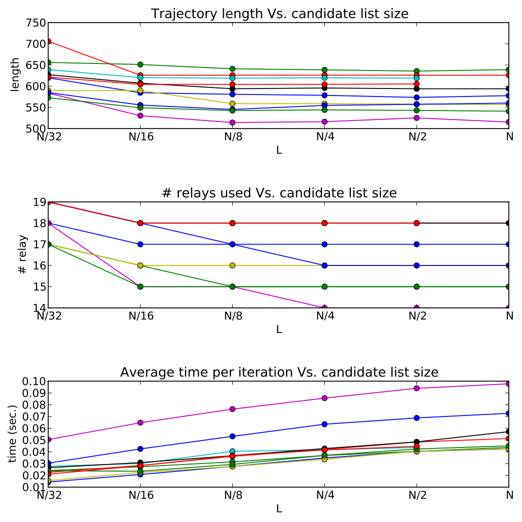


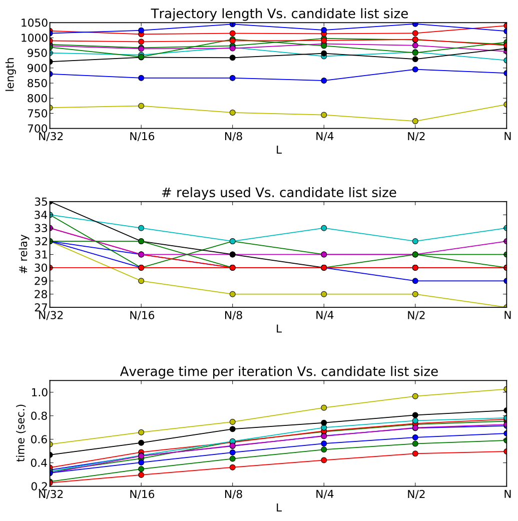
Figure 4.13.: Comparison different strategies for calculating potential relay locations on networks with $n = 50$.

4.6. Size of Candidate List

During the construction of a tentative solution, each ant maintains a list of most attractive unvisited potential relay locations. This list is called the *candidate list* [26, 22]. The size of the candidate list is denoted by L. Experiments were performed by varying the value of L To find out the effect of candidate list size on the output. The value of L was varied within the set $\{N, \frac{N}{2}, \frac{N}{4}, \frac{N}{8}, \frac{N}{16}, \frac{N}{32}, \frac{N}{64}\}$. The trajectory length, number of relays used and average running time per iteration are shown as semi-log plots in Figure 4.14. For all networks, the average running time per iteration increased logarithmically with respect to L. For networks with $n = 25$, an optimum value of the trajectory length, and the number of relays used was found for $L \geq \frac{N}{4}$. In case of networks with $n = 50$, a similar phenomena was also observed, with some fluctuations.

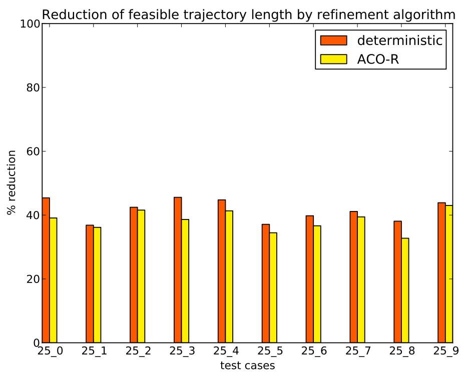


(a) Networks with $n = 25$.

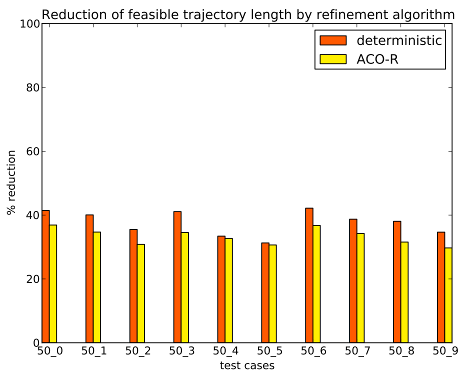


(b) Networks with $n = 50$

Figure 4.14.: Effect of the candidate list size on the output.



(a) Reduction of the feasible trajectory lengths in networks with $n = 25$



(b) Reduction of the feasible trajectory lengths in networks with $n = 50$

Figure 4.15.: Comparison of trajectory refinement algorithms.

4.7. Comparison of Trajectory Refinement Algorithms

The two trajectory refinement approaches presented in Chapter 3 were compared by calculating the length difference between the feasible trajectory produced in Phase-1, and the refined trajectory produced in Phase-2. The reduction of length was expressed as a percentage of the feasible trajectory. The results are listed in Table A.9, Table A.10 and illustrated in Figure 4.15. The deterministic algorithm performed better than the $\mathrm{ACO}_{\mathbb{R}}$ algorithm for all the networks.

5. Conclusion & Future Work

5.1. Conclusion

In this thesis, a new Ant Colony Optimization (ACO) approach for solving the joint problem of Relay Node Placement and Trajectory calculation (RNPT) has been proposed. This is the first ACO based approach for solving the RNPT problem, to the best of the author's knowledge. Given the locations of the sensor nodes in a sensing field, the proposed approach calculates a set of locations for placing relay nodes, and an optimized trajectory for the MDC for visiting the said relay nodes such that:

1. each sensor is covered by at least one relay;
2. the number of relays is minimized;
3. the length of the trajectory is minimized.

Several ACO variants were compared to find out the best suited one for our problem. The results were also compared with that produced by the greedy approach. Results produced by the MMAS and EAS were comparable to that produced by the greedy approach. In terms of the number of relays used and the length of the resulting trajectory, MMAS was found to produce the best results while EAS was found to produce good results within a shorter running time. However, if the priority of minimizing the number of relays can be relaxed, then using EAS is recommended because of shorter running time without compromising the solution quality.

Experiments were performed to compare two different cost metrics, and two different heuristic information for the proposed ACO approach. Using a cost metric specialized for the RNPT problem along with the heuristic information proposed for solving TSP by other researchers produced the best results.

A deterministic, and a Continuous Ant Colony Optimization ($ACO_{\mathbb{R}}$) approach for further refining the trajectory produced by the ACO approach have been proposed in this thesis. These trajectory refinement approaches are in fact, algorithms for solving the TSPN problem in non-sparse neighborhoods. The deterministic approach was found to perform better than the $ACO_{\mathbb{R}}$ approach, but the results were comparable.

5.2. Future Work

The work presented in this thesis can be extended in the following ways:

- Designing an appropriate heuristic information for the ACO approach: The effect of using a specialized cost metric was well observed in the reported experiments. There is potential for further improvement if an appropriate heuristic information is used.
- Designing an appropriate local search for the ACO approach: Experiments were conducted using a local search technique[26] suggested for the TSP problem. Using a customized local search for the RNPT problem is likely to improve the performence.
- Incorporating realistic sensing field: The sensing field has been assumed to be flat and devoid of obstacles. In real world application, the sensing field is

often irregular and obstacles are common. Therefore, modifying the proposed approach to address the presence of obstacles in an irregular sensing field is of practical interest.

- Modifying the proposed $ACO_{\mathbb{R}}$ approach for trajectory refinement: The performance of the proposed $ACO_{\mathbb{R}}$ approach for trajectory refinement was comparable to that of the deterministic algorithm. Further research can be carried out to modify the $ACO_{\mathbb{R}}$ approach to improve its performance.

A. Appendix

Network	# relays	Trajectory length
25_0	628.702	17
25_1	744.261	15
25_2	731.606	18
25_3	723.528	18
25_4	510.774	14
25_5	586.061	16
25_6	647.145	18
25_7	552.734	17
25_8	604.636	15
25_9	673.611	18

Table A.1.: Greedy solutions for the networks with $n = 25$.

Network	# relays	Trajectory length
50_0	1025.017	28
50_1	998.617	30
50_2	1076.676	30
50_3	1018.752	32
50_4	1122.001	31
50_5	845.705	29
50_6	1062.896	28
50_7	1173.050	30
50_8	1099.817	31
50_9	1129.523	30

Table A.2.: Greedy solutions for the networks with $n = 50$.

Network	Performance metric	ACO Approach		
		AS	EAS	MMAS
25_0	Feasible trajectory(CV)	595.389(0.022)	580.763(0.005)	577.739(0.006)
	# Relays(CV)	17.0(0.0)	17.0(0.0)	17.0(0.0)
25_1	Feasible trajectory(CV)	645.331(0.011)	642.719(0.01)	638.16(0.003)
	# Relays(CV)	15.0(0.0)	15.0(0.0)	15.0(0.0)
25_2	Feasible trajectory(CV)	643.165(0.034)	640.781(0.019)	626.054(0.0)
	# Relays(CV)	18.0(0.0)	18.0(0.0)	18.0(0.0)
25_3	Feasible trajectory(CV)	622.702(0.006)	637.55(0.019)	619.059(0.0)
	# Relays(CV)	18.0(0.0)	18.0(0.0)	18.0(0.0)
25_4	Feasible trajectory(CV)	514.406(0.006)	510.142(0.035)	514.848(0.004)
	# Relays(CV)	14.0(0.0)	14.2(0.031)	14.0(0.0)
25_5	Feasible trajectory(CV)	557.991(0.003)	526.307(0.007)	558.63(0.003)
	# Relays(CV)	16.0(0.0)	17.0(0.0)	16.0(0.0)
25_6	Feasible trajectory(CV)	597.898(0.006)	559.84(0.023)	593.953(0.0)
	# Relays(CV)	18.0(0.0)	19.2(0.023)	18.0(0.0)
25_7	Feasible trajectory(CV)	552.323(0.015)	502.54(0.021)	554.955(0.011)
	# Relays(CV)	16.0(0.0)	17.8(0.025)	16.0(0.0)
25_8	Feasible trajectory(CV)	542.797(0.004)	544.363(0.003)	541.897(0.002)
	# Relays(CV)	15.0(0.0)	15.0(0.0)	15.0(0.0)
25_9	Feasible trajectory(CV)	609.566(0.011)	611.646(0.012)	604.403(0.001)
	# Relays(CV)	18.0(0.0)	18.0(0.0)	18.0(0.0)

Table A.3.: Performence of ACO approaches on networks with $n = 25$, using cost metric C^1.

Network	Performance metric	ACO Approach		
		AS	EAS	MMAS
50_0	Feasible trajectory(CV)	934.978(0.023)	910.888(0.026)	893.081(0.018)
	# Relays(CV)	31.2(0.027)	29.6(0.045)	29.4(0.019)
50_1	Feasible trajectory(CV)	1008.509(0.022)	1029.638(0.037)	992.222(0.016)
	# Relays(CV)	32.4(0.035)	31.8(0.014)	31.2(0.014)
50_2	Feasible trajectory(CV)	1083.943(0.035)	1034.744(0.023)	1010.792(0.019)
	# Relays(CV)	32.0(0.0)	31.4(0.017)	30.6(0.018)
50_3	Feasible trajectory(CV)	1022.008(0.04)	1002.547(0.035)	964.868(0.018)
	# Relays(CV)	33.2(0.013)	33.0(0.021)	32.2(0.014)
50_4	Feasible trajectory(CV)	1047.533(0.032)	1000.021(0.013)	971.92(0.018)
	# Relays(CV)	32.0(0.038)	31.0(0.0)	31.0(0.0)
50_5	Feasible trajectory(CV)	783.39(0.041)	771.813(0.01)	758.426(0.028)
	# Relays(CV)	29.6(0.07)	27.6(0.032)	28.2(0.016)
50_6	Feasible trajectory(CV)	984.29(0.057)	956.776(0.033)	961.135(0.014)
	# Relays(CV)	30.8(0.053)	30.8(0.042)	30.4(0.029)
50_7	Feasible trajectory(CV)	1065.273(0.013)	1050.847(0.028)	1034.046(0.015)
	# Relays(CV)	32.0(0.022)	30.2(0.015)	30.2(0.015)
50_8	Feasible trajectory(CV)	990.465(0.027)	983.255(0.014)	975.777(0.019)
	# Relays(CV)	30.2(0.015)	30.0(0.0)	30.8(0.015)
50_9	Feasible trajectory(CV)	1017.706(0.03))	1023.766(0.026)	986.779(0.006)
	# Relays(CV)	30.2(0.015	30.0(0.0)	30.0(0.0)

Table A.4.: Performence of ACO approaches on networks with $n = 50$, using cost metric C^1.

Network	Performance metric	ACO Approach		
		AS	EAS	MMAS
25_0	Feasible trajectory(CV)	610.76(0.021)	576.40(0.007)	574.207(0.002)
	# Relays(CV)	17.2(0.026)	17(0.000)	17.6(0.031)
25_1	Feasible trajectory(CV)	652.17(0.010)	640.186(0.030)	635.693(0.000)
	# Relays(CV)	16.2(0.080)	18(0.000)	16.6(0.033)
25_2	Feasible trajectory(CV)	652.85(0.036)	635.285(0.000)	625.916(0.000)
	# Relays(CV)	18.8(0.044)	15(0.000)	18(0.000)
25_3	Feasible trajectory(CV)	651.93(0.028)	619.059(0.000)	619.103(0.000)
	# Relays(CV)	18.4(0.030)	18(0.000)	18.2(0.024)
25_4	Feasible trajectory(CV)	517.13(0.002)	510.490(0.001)	512.224(0.001)
	# Relays(CV)	16.4(0.054)	14.2(0.0315)	15(0.067)
25_5	Feasible trajectory(CV)	573.400(0.015)	556.914(0.003)	556.904(0.001)
	# Relays(CV)	17(0.041)	17(0.000)	17.2(0.049)
25_6	Feasible trajectory(CV)	618.547(0.030)	593.571(0.001)	593.479(0.000)
	# Relays(CV)	18.6(0.048)	19.2(0.0233)	19(0.000)
25_7	Feasible trajectory(CV)	555.378(0.011)	542.136(0.001)	542.197(0.001)
	# Relays(CV)	18(0.039)	17.8(0.025)	17.8(0.047)
25_8	Feasible trajectory(CV)	547.262(0.006)	540.150(0.000)	541.171(0.000)
	# Relays(CV)	16.6(0.081)	15(0.000)	15.8(0.028)
25_9	Feasible trajectory(CV)	634.914(0.031)	603.436(0.000)	604.177(0.000)
	# Relays(CV)	18.2(0.024)	18(0.000)	18(0.000)

Table A.5.: Performence of ACO approaches on networks with $n = 25$, using cost metric C^2.

Network	Performance metric	ACO Approach		
		AS	EAS	MMAS
50_0	Feasible trajectory(CV)	946.397(0.016)	844.521(0.020)	847.917(0.007)
	# Relays(CV)	32.6(0.027)	29.4(0.019)	34.2(0.0131)
50_1	Feasible trajectory(CV)	1081.105(0.043)	934.861(0.005)	952.371(0.002)
	# Relays(CV)	35.6(0.047)	30.6(0.0292)	35.2(0.047)
50_2	Feasible trajectory(CV)	1073.863(0.0143)	962.001(0.008)	977.968(0.006)
	# Relays(CV)	32.6(0.041)	30(0.000)	33.8(0.0248)
50_3	Feasible trajectory(CV)	1041.056(0.0462)	922.564(0.010)	931.623(0.004)
	# Relays(CV)	35.8(0.023)	32.2(0.014)	35.2(0.031)
50_4	Feasible trajectory(CV)	1013.0305(0.0146)	952.542(0.015)	954.387(0.004)
	# Relays(CV)	33.8(0.038)	31(0.000)	33(0.037)
50_5	Feasible trajectory(CV)	842.111(0.037)	699.741(0.012)	721.624(0.007)
	# Relays(CV)	31.8(0.060)	26.4(0.021)	31.4(0.048)
50_6	Feasible trajectory(CV)	1006.643(0.016)	905.788(0.011)	913.949(0.008)
	# Relays(CV)	35.6(0.055)	28.8(0.029)	34(0.036)
50_7	Feasible trajectory(CV)	1096.312(0.055)	988.632(0.003)	997.794(0.006)
	# Relays(CV)	33.2(0.025)	30.4(0.018)	34.2(0.013)
50_8	Feasible trajectory(CV)	1072.507(0.037)	928.194(0.000)	948.030(0.005)
	# Relays(CV)	32.2(0.026)	30(0.000)	32.8(0.040)
50_9	Feasible trajectory(CV)	1058.579(0.030)	981.179(0.021)	977.812(0.003)
	# Relays(CV)	31(0.032)	30.6(0.018)	31.4(0.043)

Table A.6.: Performence of ACO approaches on networks with $n = 50$, using cost metric C^2

Network	Performance Metric	Strategy		
		Centre	4 points	8 points
25_0	Feasible Trajectory(CV)	893.081(0.018)	533.935(0.007)	527.826(0.007)
	Refined Trajectory(CV)	315.409(0.032)	326.383(0.034)	313.815(0.028)
	# potential relay location	32	56	96
25_1	Feasible Trajectory(CV)	992.222(0.016)	535.285(0.007)	534.937(0.011)
	Refined Trajectory(CV)	403.065(0.005)	410.268(0.013)	401.153(0.008)
	# potential relay location	35	53	83
25_2	Feasible Trajectory(CV)	1010.792(0.019)	599.442(0.023)	603.331(0.012)
	Refined Trajectory(CV)	360.271(0.010)	369.909(0.033)	359.878(0.020)
	# potential relay location	33	66	121
25_3	Feasible Trajectory(CV)	964.868(0.018)	580.183(0.016)	578.418(0.015)
	Refined Trajectory(CV)	336.960(0.018)	348.519(0.006)	339.973(0.009)
	# potential relay location	30	60	110
25_4	Feasible Trajectory(CV)	971.92(0.018)	398.684(0.013)	398.689(0.035)
	Refined Trajectory(CV)	284.354(0.012)	298.998(0.035)	296.832(0.030)
	# potential relay location	61	82	117
25_5	Feasible Trajectory(CV)	758.426(0.028)	489.536(0.016)	491.995(0.006)
	Refined Trajectory(CV)	351.331(0.039)	348.037(0.008)	348.243(0.017)
	# potential relay location	35	56	91
25_6	Feasible Trajectory(CV)	961.135(0.014)	550.518(0.003)	545.664(0.011)
	Refined Trajectory(CV)	357.612(0.013)	337.861(0.024)	344.892(0.025)
	# potential relay location	32	68	128
25_7	Feasible Trajectory(CV)	1034.046(0.015)	484.554(0.026)	477.822(0.022)
	Refined Trajectory(CV)	326.734(0.039)	345.305(0.063)	341.906(0.055)
	# potential relay location	46	73	118
25_8	Feasible Trajectory(CV)	975.777(0.019)	447.080(0.013)	447.907(0.011)
	Refined Trajectory(CV)	335.459(0.010)	356.925(0.018)	352.870(0.030)
	# potential relay location	39	54	79
25_9	Feasible Trajectory(CV)	986.779(0.006)	554.593(0.008)	570.796(0.066)
	Refined Trajectory(CV)	339.246(0.043)	328.033(0.018)	335.769(0.031)
	# potential relay location	31	64	119

Table A.7.: Comparing different strategies for calculating potential relay locations on networks with $n = 25$.

Network	Performance Metric	Strategy		
		Centre	4 points	8 points
50_0	Feasible Trajectory	893.081(0.018)	845.784(0.039)	819.828(0.000)
	Refined Trajectory	523.124(0.088)	537.715(0.092)	0.303(0.014)
	# potential relay location	85	109	149
50_1	Feasible Trajectory	992.221(0.016)	913.693(0.021)	931.992(0.002)
	Refined Trajectory	594.628(0.029)	611.708(0.040)	0.441(0.009)
	# potential relay location	78	114	174
50_2	Feasible Trajectory	1010.792(0.019)	981.508(0.023)	949.844(0.000)
	Refined Trajectory	651.828(0.013)	718.031(0.033)	0.399(0.023)
	# potential relay location	83	116	171
50_3	Feasible Trajectory	964.868(0.018)	923.884(0.017)	914.595(0.000)
	Refined Trajectory	568.222(0.057)	590.885(0.046)	0.445(0.023)
	# potential relay location	79	115	175
50_4	Feasible Trajectory	971.920(0.018)	920.641(0.023)	938.248(0.000)
	Refined Trajectory	647.121(0.0239)	639.081(0.047)	0.378(0.008)
	# potential relay location	81	114	169
50_5	Feasible Trajectory	758.425(0.028)	761.320(0.088)	702.855(0.005)
	Refined Trajectory	521.092(0.060)	508.138(0.050)	0.297(0.013)
	# potential relay location	120	132	152
50_6	Feasible Trajectory	961.135(0.014)	908.088(0.038)	895.655(0.002)
	Refined Trajectory	555.774(0.039)	572.453(0.088)	0.418(0.014)
	# potential relay location	91	179	179
50_7	Feasible Trajectory	1034.046(0.015)	976.886(0.022)	986.951(0.002)
	Refined Trajectory	633.601(0.027)	654.955(0.051)	0.401(0.008)
	# potential relay location	73	177	177
50_8	Feasible Trajectory	975.777(0.019)	934.952(0.033)	929.079(0.001)
	Refined Trajectory	604.334(0.027)	625.773(0.074)	0.365(0.010)
	# potential relay location	70	168	166
50_9	Feasible Trajectory	986.779(0.005)	956.382(0.011)	963.869(0.000)
	Refined Trajectory	644.773(0.013)	659.359(0.035)	0.274(0.016)
	# potential relay location	68	148	148

Table A.8.: Comparing different strategies for calculating potential relay locations on networks with $n = 50$.

Network	% Reduction of trajectory length	
	Deterministic	$ACO_{\mathbb{R}}$
25_0	45.40	39.10
25_1	36.84	36.16
25_2	42.45	41.56
25_3	45.57	38.63
25_4	44.77	41.34
25_5	37.11	34.46
25_6	39.79	36.65
25_7	41.14	39.44
25_8	38.09	32.75
25_9	43.87	43.02

Table A.9.: Comparison of trajectory refinement algorithms on networks with $n = 25$.

Network	% Reduction of trajectory length	
	Deterministic	$ACO_{\mathbb{R}}$
25_0	41.46	36.88
25_1	40.06	34.69
25_2	35.50	30.83
25_3	41.09	34.56
25_4	33.41	32.70
25_5	31.28	30.64
25_6	42.17	36.77
25_7	38.71	34.26
25_8	38.06	31.55
25_9	34.65	29.72

Table A.10.: Comparison of trajectory refinement algorithms on networks with $n = 50$.

Bibliography

[1] Confidence intervals for the coefficient of variation for the normal and log normal distributions. *Biometrika 51* (1964), 25–32.

[2] Catalogue wireless sensor networks 2013, 2013.

[3] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Comput. Netw. 38*, 4 (Mar. 2002), 393–422.

[4] Arkin, E. M., and Hassin, R. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics 55*, 3 (1994), 197 – 218.

[5] Baker, C., Armijo, K., Belka, S., Benhabib, M., Bhargava, V., Burkhart, N., Der Minassians, A., Dervisoglu, G., Gutnik, L., Haick, M., Ho, C., Koplow, M., Mangold, J., Robinson, S., Rosa, M., Schwartz, M., Sims, C., Stoffregen, H., Waterbury, A., Leland, E., Pering, T., and Wright, P. Wireless sensor networks for home health care. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on* (2007), vol. 2, pp. 832–837.

[6] Balaprakash, P., Birattari, M., StÃŒtzle, T., Yuan, Z., and Dorigo, M. Estimation-based ant colony optimization and local search for

the probabilistic traveling salesman problem. *Swarm Intelligence 3*, 3 (2009), 223–242.

[7] Bansal, N., Blum, A., Chawla, S., and Meyerson, A. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 2004), STOC '04, ACM, pp. 166–174.

[8] Bari, A., Chen, Y., Roy, D., Jaekel, A., and Bandyopadhyay, S. Designing hierarchical sensor networks with mobile data collectors. *Pervasive and Mobile Computing 7*, 1 (2011), 128 – 139.

[9] Bari, A., and Jaekel, A. Techniques for exploiting mobility in wireless sensor networks. *Handbook of Research in Mobile Business, Second Edition: Technical, Methodological and Social Perspectives* (2009), 445–455.

[10] Bari, A., Jaekel, A., and Bandyopadhyay, S. Integrated clustering and routing strategies for large scale sensor networks. In *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, I. Akyildiz, R. Sivakumar, E. Ekici, J. Oliveira, and J. McNair, Eds., vol. 4479 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 143–154.

[11] Bari, A., Jaekel, A., and Bandyopadhyay, S. Clustering strategies for improving the lifetime of two-tiered sensor networks. *Computer Communications 31*, 14 (2008), 3451 – 3459.

[12] Bari, A., Jaekel, A., Jiang, J., and Xu, Y. Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements. *Computer Communications 35*, 3 (2012), 320 – 333.

[13] Bland, R. G., and Shallcross, D. F. Large travelling salesman problems

arising from experiments in x-ray crystallography: A preliminary report on computation. *Operations Research Letters 8*, 3 (1989), 125 – 128.

[14] Bonabeau, E., Theraulaz, G., Deneubourg, J.-L., Aron, S., and Camazine, S. Self-organization in social insects. *Trends in Ecology & Evolution 12*, 5 (1997), 188 – 193.

[15] Cayirci, E., and Coplu, T. Sendrom: Sensor networks for disaster relief operations management. *Wireless Networks 13*, 3 (2007), 409–423.

[16] Chong, C.-Y., and Kumar, S. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE 91*, 8 (2003), 1247–1256.

[17] Comarela, G., GonÃ§alves, K. C., Pappa, G. L., Almeida, J. M., and Almeida, V. Robot routing in sparse wireless sensor networks with continuous ant colony optimization. In *GECCO (Companion)* (2011), N. Krasnogor and P. L. Lanzi, Eds., ACM, pp. 599–606.

[18] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, 2 ed. The MIT Press, 2001.

[19] Costa, D., and Hertz, A. Ants can colour graphs. In *The Journal of the Operational Research Society*, vol. 48-3. Palgrave Macmillan Journals, 1997, pp. 295–305.

[20] de Berg, M., Gudmundsson, J., Katz, M. J., Levcopoulos, C., Overmars, M. H., and van der Stappen, A. F. {TSP} with neighborhoods of varying size. *Journal of Algorithms 57*, 1 (2005), 22 – 36.

[21] Dorigo, M. *Optimization, Learning and Natural Algorithms [in Italian]*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, 1992.

[22] Dorigo, M., and Gambardella, L. Ant colony system: a cooperative

learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on 1*, 1 (1997), 53–66.

[23] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: An autocatalytic optimizing process, technical report 91-016 revised. Tech. rep., Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.

[24] Dorigo, M., Maniezzo, V., and Colorni, A. Positive feedback as a search strategy, technical report 91-016 revised. Tech. rep., Dipartimento di Elettronica, Politecnico di Milano, Milan, 1991.

[25] Dorigo, M., Maniezzo, V., and Colorni, A. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26*, 1 (1996), 29–41.

[26] Dorigo, M., and St utzle, T., Eds. *Ant Colony Optimization.* The MIT Press, Cambridge, Massachusetts, 2004.

[27] Force, U. S. Sosus the secret weapon of undersea surveillance, 2005.

[28] Fowler, R. J., Paterson, M. S., and Tanimoto, S. L. Optimal packing and covering in the plane are np-complete. *Information Processing Letters 12*, 3 (1981), 133 – 137.

[29] Gandham, S., Dawande, M., Prakash, R., and Venkatesan, S. Energy efficient schemes for wireless sensor networks with multiple mobile base stations. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE* (2003), vol. 1, pp. 377–381 Vol.1.

[30] Garey, M. R., Graham, R. L., and Johnson, D. S. Some np-complete geometric problems. In *Proceedings of the eighth annual ACM symposium on Theory of computing* (New York, NY, USA, 1976), STOC '76, ACM, pp. 10–22.

[31] Garfinkel, R. S. Minimizing wallpaper waste, part 1: A class of traveling salesman problems. *Operations Research 25*, 5 (1977), pp. 741–751.

[32] Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. Self-organized shortcuts in the argentine ant. *Naturwissenschaften 76*, 12 (1989), 579–581.

[33] Gottlieb, J., Puchta, M., and Solnon, C. A study of greedy, local search, and ant colony optimization approaches for car sequencing problems. In *Applications of Evolutionary Computing*, S. Cagnoni, C. Johnson, J. Cardalda, E. Marchiori, D. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G. Raidl, and E. Hart, Eds., vol. 2611 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 246–257.

[34] Gu, Y., Bozdag, D., Ekici, E., Özgüner, F., and Lee, C.-G. Partitioning based mobile element scheduling in wireless sensor networks. In *IN. PROC. SECOND ANNUAL IEEE CONFERENCE ON SENSOR AND AD HOC COMMUNICATIONS AND NETWORKS (SECON* (2005), pp. 386–395.

[35] Gungor, V., and Hancke, G. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on 56*, 10 (2009), 4258–4265.

[36] Gupta, G., and Younis, M. Load-balanced clustering of wireless sensor networks. In *Communications, 2003. ICC '03. IEEE International Conference on* (2003), vol. 3, pp. 1848–1852 vol.3.

[37] Hansen, N., and Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput. 9*, 2 (June 2001), 159–195.

[38] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. In *System*

Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on (2000), pp. 10 pp. vol.2–.

[39] Hou, Y., Shi, Y., Sherali, H., and Midkiff, S. On energy provisioning and relay node placement for wireless sensor networks. *Wireless Communications, IEEE Transactions on 4*, 5 (2005), 2579–2590.

[40] Howard, A., Matarić, M., and Sukhatme, G. An incremental self-deployment algorithm for mobile sensor networks. *Autonomous Robots 13*, 2 (2002), 113–126.

[41] Howard, A., Matarić, M., and Sukhatme, G. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Eds. Springer Japan, 2002, pp. 299–308.

[42] Ihler, E., Reich, G., and Widmayer, P. On shortest networks for classes of points in the plane. In *Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications* (London, UK, UK, 1991), CG '91, Springer-Verlag, pp. 103–111.

[43] Isler, V., Kannan, S., and Daniilidis, K. Sampling based sensor-network deployment. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (2004), vol. 2, pp. 1780–1785 vol.2.

[44] Jain, S., Shah, R. C., Brunette, W., Borriello, G., and Roy, S. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mob. Netw. Appl. 11*, 3 (June 2006), 327–339.

[45] Jea, D., Somasundara, A., and Srivastava, M. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Dis-*

tributed Computing in Sensor Systems, V. Prasanna, S. Iyengar, P. Spirakis, and M. Welsh, Eds., vol. 3560 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 244–257.

[46] Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., and Koumoutsakos, P. Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing 3*, 1 (2004), 77–112.

[47] Keskin, M. E., Altinel, İ. K., Aras, N., and Ersoy, C. Lifetime maximization in wireless sensor networks using a mobile sink with nonzero traveling time. *The Computer Journal 54*, 12 (2011), 1987–1999.

[48] Laporte, G. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research 59*, 2 (1992), 231 – 247.

[49] Lenstra, J. K., and Kan, A. H. G. R. Some simple applications of the travelling salesman problem. *Operational Research Quarterly (1970-1977) 26*, 4 (1975), pp. 717–733.

[50] Lessing, L., Dumitrescu, I., and StÃŒtzle, T. A comparison between aco algorithms for the set covering problem. In *Ant Colony Optimization and Swarm Intelligence*, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. StÃŒtzle, Eds., vol. 3172 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–12.

[51] Levine, J., and Ducatelle, F. Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society. (forthcoming 93* (2003), 2003.

[52] Lin, R., Wang, Z., and Sun, Y. Wireless sensor networks solutions for real

time monitoring of nuclear power plant. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on* (2004), vol. 4, pp. 3663–3667 Vol.4.

[53] Liu, J.-S., Wu, S.-Y., and Chiu, K.-M. Path planning of a data mule in wireless sensor network using an improved implementation of clustering-based genetic algorithm. In *Computational Intelligence in Control and Automation (CICA), 2013 IEEE Symposium on* (2013), pp. 30–37.

[54] Lloyd, E., and Xue, G. Relay node placement in wireless sensor networks. *Computers, IEEE Transactions on 56*, 1 (2007), 134–138.

[55] Luo, J., and Hubaux, J.-P. Joint mobility and routing for lifetime elongation in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* (2005), vol. 3, pp. 1735–1746 vol. 3.

[56] Luo, J., and Hubaux, J.-P. Joint sink mobility and routing to maximize the lifetime of wireless sensor networks: The case of constrained mobility. *Networking, IEEE/ACM Transactions on 18*, 3 (2010), 871–884.

[57] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (New York, NY, USA, 2002), WSNA '02, ACM, pp. 88–97.

[58] Martin, O., Otto, S. W., and Felten, E. W. Large-step markov chains for the {TSP} incorporating local search heuristics. *Operations Research Letters 11*, 4 (1992), 219 – 224.

[59] Mata, C. S., and Mitchell, J. S. B. Approximation algorithms for geometric tour and network design problems (extended abstract). In *Proceedings*

of the eleventh annual symposium on Computational geometry (New York, NY, USA, 1995), SCG '95, ACM, pp. 360–369.

[60] MEDICINENET.COM. Definition of pheromone, 2012.

[61] MISRA, S., HONG, S. D., XUE, G., AND TANG, J. Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE* (2008), pp. –.

[62] NAKAYAMA, H., ANSARI, N., JAMALIPOUR, A., AND KATO, N. Fault-resilient sensing in wireless sensor networks. *Computer Communications: Special issue on security on wireless ad hoc and sensor networks 30*, 11 - 12 (2007), 2375 – 2384.

[63] OSMAN, I., AND LAPORTE, G. Metaheuristics: A bibliography. *Annals of Operations Research 63*, 5 (1996), 511–623.

[64] OSTERMEIER, A., GAWELCZYK, A., AND HANSEN, N. Step-size adaptation based on non-local use of selection information. In *Parallel Problem Solving from Nature – PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., vol. 866 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1994, pp. 189–198.

[65] RAHMAN, M., AND NAZNIN, M. Shortening the tour-length of a mobile data collector in the wsn by the method of linear shortcut. In *Web Technologies and Applications*, Y. Ishikawa, J. Li, W. Wang, R. Zhang, and W. Zhang, Eds., vol. 7808 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 674–685.

[66] REICH, G., AND WIDMAYER, P. Beyond steiner's problem: a vlsi oriented generalization. In *Proceedings of the fifteenth international workshop on Graph-*

theoretic concepts in computer science (New York, NY, USA, 1990), WG '89, Springer-Verlag New York, Inc., pp. 196–210.

[67] Rodgers, J. L., and Nicewander, W. A. Thirteen ways to look at the correlation coefficient. In *The American Statistician*, vol. 42-1. American Statistical Association, 1988, pp. 59–66.

[68] Romer, K., and Mattern, F. The design space of wireless sensor networks. *Wireless Communications, IEEE 11*, 6 (2004), 54–61.

[69] Rosenkrantz, D., Stearns, R., and Lewis, PhilipM., I. An analysis of several heuristics for theÂ traveling salesman problem. In *Fundamental Problems in Computing*, S. Ravi and S. Shukla, Eds. Springer Netherlands, 2009, pp. 45–69.

[70] Safra, S., and Schwartz, O. On the complexity of approximating tsp with neighborhoods and related problems. *computational complexity 14*, 4 (2006), 281–307.

[71] Schwefel, H.-P. P. *Evolution and Optimum Seeking: The Sixth Generation.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[72] Shah, R., Roy, S., Jain, S., and Brunette, W. Data mules: modeling a three-tier architecture for sparse sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on* (2003), pp. 30–41.

[73] Socha, K., and Dorigo, M. Ant colony optimization for continuous domains. *European Journal of Operational Research 185*, 3 (2008), 1155 – 1173.

[74] Stutzle, T., and Hoos, H. Max-min ant system and local search for the traveling salesman problem. In *Evolutionary Computation, 1997., IEEE International Conference on* (1997), pp. 309–314.

[75] Stützle, T., and Hoos, H. H. Max-min ant system. *Future Generation Computer Systems 16*, 8 (2000), 889 – 914.

[76] Suomela, J. Computational complexity of relay placement in sensor networks. In *SOFSEM 2006: Theory and Practice of Computer Science*, J. Wiedermann, G. Tel, J. PokornÃœ, M. BielikovÃ¡, and J. Å tuller, Eds., vol. 3831 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 521–529.

[77] Szewczyk, R., Osterweil, E., Polastre, J., Hamilton, M., Mainwaring, A., and Estrin, D. Habitat monitoring with sensor networks. *Commun. ACM 47*, 6 (June 2004), 34–40.

[78] Tang, J., Hao, B., and Sen, A. Relay node placement in large scale wireless sensor networks. *Comput. Commun. 29*, 4 (Feb. 2006), 490–501.

[79] Wan, P.-J., and Yi, C.-W. Coverage by randomly deployed wireless sensor networks. In *Network Computing and Applications, Fourth IEEE International Symposium on* (2005), pp. 275–278.

[80] Wang, Y. Topology control for wireless sensor networks. In *Wireless Sensor Networks and Applications*, Y. Li, M. Thai, and W. Wu, Eds., Signals and Communication Technology. Springer US, 2008, pp. 113–147.

[81] Wang, Z., Basagni, S., Melachrinoudis, E., and Petrioli, C. Exploiting sink mobility for maximizing sensor networks lifetime. In *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on* (2005), pp. 287a–287a.

[82] Wark, T., Corke, P., Sikka, P., Klingbeil, L., Guo, Y., Crossman, C., Valencia, P., Swain, D., and Bishop-Hurley, G. Transforming agriculture through pervasive wireless sensor networks. *Pervasive Computing, IEEE 6*, 2 (2007), 50–57.

[83] Weisstein, E. W. Np-hard problem, 2013. [Online; accessed 22-November-2013].

[84] Weisstein, E. W. Np-problem, 2013. [Online; accessed 22-November-2013].

[85] Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., and Welsh, M. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on* (2005), pp. 108–120.

[86] Wu, C.-H., Lee, K.-C., and Chung, Y.-C. A delaunay triangulation based method for wireless sensor network deployment. *Computer Communications 30*, 14 - 15 (2007), 2744 – 2752. Network Coverage and Routing Schemes for Wireless Sensor Networks.

[87] Xu, Y.-L., Lim, M.-H., Ong, Y.-S., and Tang, J. A ga-aco-local search hybrid algorithm for solving quadratic assignment problem. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2006), GECCO '06, ACM, pp. 599–606.

[88] Yuan, B., Orlowska, M., and Sadiq, S. On the optimal robot routing problem in wireless sensor networks. *Knowledge and Data Engineering, IEEE Transactions on 19*, 9 (2007), 1252–1261.

[89] Zheng, L. Zigbee wireless sensor network in industrial applications. In *SICE-ICASE, 2006. International Joint Conference* (2006), pp. 1067–1070.

[90] Zhou, B., Gao, L., and Dai, Y.-H. Gradient methods with adaptive step-sizes. *Comput. Optim. Appl. 35*, 1 (Sept. 2006), 69–86.

VITA AUCTORIS

Name	K Raiyan Kamal
Birth	1985
	Rajshahi, Bangladesh
Education	
	Notre Dame College, Dhaka, Bangladesh
	HSC 2003
	Bangladesh University of Engineering & Technology, Dhaka, Bangladesh
	BSc. Computer Science & Engineering, 2009
	University of Windsor, Ontario, Canada
	MSc. Computer Science 2014