HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Evolutionary and Deep Reinforcement Learning Algorithms for Optimizing the Lifetime of Wireless Sensor Networks

BUI HONG NGOC
ngoc.bh212155m@sis.hust.edu

Major : Data Science (Elitech)

Thesis advisor	:	Dr. Nguyen Phi Le
Department	:	Department of Computer Science
Institute	:	School of Information and Communication Technology

Hanoi, 4-2023

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Evolutionary and Deep Reinforcement Learning Algorithms for Optimizing the Lifetime of Wireless Sensor Networks

BUI HONG NGOC
ngoc.bh212155m@sis.hust.edu

Major : Data Science (Elitech)

Thesis advisor : Dr. Nguyen Phi Le

Signature of advisor

Department : Department of Computer Science
Institute : School of Information and Communication Technology

Hanoi, 4-2023

GRADURATION THESIS ASSIGNMENT

1. Student's information :
   Name : Bui Hong Ngoc.
   Phone : 0988 490 924 Email: ngoc.bh212155m@sis.hust.edu
   Class : Data Science
   Affiliation : Hanoi University of Science and Technology.
   Duration : 10/11/2022 - 22/04/2023.

2. Thesis title : Evolutionary and Deep Reinforcement Learning Algorithms for Optimizing the Lifetime of Wireless Sensor Networks

3. Thesis statement :
   This thesis proposes evolutionary and deep reinforcement learning algorithms to tackle two emerging techniques in prolonging the lifetime of wireless sensor networks.

4. Declarations/Disclosures :
   I – Bui Hong Ngoc – hereby warrants that the work and presentation in this thesis are performed by myself under the supervision of Dr. Nguyen Phi Le. All results presented in this thesis are truthful and are not copied from any other works. All references in this thesis - including images, tables, figures, and quotes - are clearly and fully documented in the bibliography. I will take full responsibility for even one copy that violates school regulations.

   Hanoi, date month year 2023
   Author

   Bui Hong Ngoc

5. Attestation of thesis advisor:
   I certify that I have read this thesis and that, in my opinion, it is fully adequate in scope and quality as a thesis for the degree of Master of Science.

   Hanoi, date month year 2023
   Thesis Advisor

   Dr. Nguyen Phi Le

Acknowledgments

I would like to express my gratitude to all those who have motivated and aided me in achieving this significant milestone in my life. My heartfelt appreciation goes out to my thesis advisors, Dr. Nguyen Phi Le and Prof. Do Phan Thuan, whose encouragement and guidance were invaluable in shaping the research questions and methodology. Without their supervision and insightful suggestions, completing this thesis would have been an insurmountable challenge.

I owe a debt of gratitude to my advisor at VinAI, Dr. Nguyen Viet Anh, for his unwavering support throughout my two years at VinAI. His expertise and guidance have enabled me to not only complete this thesis but also gain the knowledge and skills necessary for my future career. Beyond academic research, he was always available to provide advice and guidance on various aspects of my personal and professional development. His support has been a source of inspiration and motivation for me throughout my journey.

Additionally, The invaluable feedback and suggestions provided by Prof. Huynh Thi Thanh Binh and Dr. Nguyen Thi Tam were essential to the completion of this thesis. Their extensive expertise in the field was crucial to the development of various aspects of this work, and I am deeply grateful for their support and guidance throughout the process. Without their contributions, this thesis would not have been completed.

I also wish to acknowledge my friends at HUST and VinAI, whose unwavering support and companionship have been a source of strength and motivation throughout my time at the university and company. The memories of playing games, watching movies, and enjoying football matches are priceless and have kept me going.

Lastly, I dedicate this thesis to my family: my parents, who have always encouraged me to pursue my dreams; my girlfriend, who has stood by me during my most difficult moments, listening and empathizing with my grievances about the world; and my sister, who may sometimes be a nuisance, but always add color to my life. Together, they have shaped the person I am today.

Evolutionary and Deep Reinforcement Learning Algorithms for Optimizing the Lifetime of Wireless Sensor Networks

Abstract

In recent years, wireless sensor networks (WSNs) have become an essential part of many civilian applications, such as smart agriculture, health monitoring, and smart cities. However, the limited energy capacity of the sensors poses a significant challenge to ensuring continuous surveillance. In this thesis, we study two emerging techniques to prolong the network's lifetime: energy-efficient routing via relay node placement and routing strategies for the mobile charger in wireless rechargeable sensor networks (WRSNs).

The first problem involves optimizing the routing protocol by deploying non-sensing relay nodes (RNs). Recent research either focused on finding the minimum number of RNs required to reduce deployment costs or developing efficient routing schemes to reduce energy use. However, striking a balance between these criteria remains a significant challenge. To address this issue, we propose a multi-objective approach to constructing an efficient communication structure with the least possible number of RNs, thereby prolonging the network's lifetime while ensuring its connectivity and reliability. We conduct extensive experiments to demonstrate the effectiveness of our method and show that it provides a better trade-off compared to existing algorithms.

In the second problem, we focus on addressing the challenges faced in WRSNs, where a mobile charger can be employed to move around and charge the sensors. Existing approaches struggle to design an optimal charging path for the mobile charger while accounting for the uncertainties arising in the network, which could come from node failures or deployments. To overcome this challenge, we propose a novel charging scheme that uses a deep reinforcement learning (DRL) approach to guide the mobile charger adaptively. Our approach enables the mobile charger to adapt to spontaneous changes in the network topology. The experiments show the superiority of our model compared to existing on-demand methods in prolonging the network lifetime.

Our proposed solutions for the two problems of relay node placement and mobile charger routing can significantly prolong the network's lifespan and reduce maintenance costs. The potential for combining these two techniques in WRSNs to further enhance the network's sustainability and reliability is an interesting future research direction. Adopting these solutions, especially new advancements in deep reinforcement learning can facilitate the development of more efficient and effective WSNs, enabling us to better monitor and manage various systems and processes in our daily lives.

Các giải thuật tiến hoá và học tăng cường để tối ưu thời gian sống của mạng cảm biến không dây

Tóm tắt luận văn

Trong những năm gần đây, mạng cảm biến không dây đã trở thành một phần cần thiết trong nhiều ứng dụng dân sự, như giám sát sức khỏe, nông nghiệp và thành phố thông minh. Tuy nhiên, lượng năng lượng hạn chế được cài đặt trong các cảm biến đặt ra thách thức lớn trong việc đảm bảo dịch vụ trong các ứng dụng yêu cần giám sát liên tục và với tần suất cao. Trong luận văn này, chúng tôi nghiên cứu hai phương pháp mới để kéo dài tuổi thọ mạng: định tuyến tiết kiệm năng lượng thông qua đặt các nút trung gian và chiến lược định tuyến cho bộ sạc di động trong mạng cảm biến có thể sạc lại không dây.

Bài toán đầu tiên liên quan đến tối ưu hóa giao thức định tuyến bằng cách triển khai các nút trung gian không cảm biến. Các nghiên cứu gần đây tập trung vào tìm số lượng tối thiểu nút trung gian cần thiết để giảm chi phí triển khai hoặc phát triển các chiến lược định tuyến để giảm năng lượng sử dụng. Tuy nhiên, tìm sự cân bằng giữa các tiêu chí này vẫn là một thách thức lớn. Để giải quyết vấn đề này, chúng tôi đề xuất một phương pháp đa mục tiêu để xây dựng một cấu trúc truyền thông hiệu quả với số lượng nút trung gian ít nhất có thể, từ đó kéo dài tuổi thọ mạng trong khi đảm bảo kết nối của nó. Chúng tôi tiến hành các thử nghiệm để chứng minh tính hiệu quả của phương pháp của chúng tôi và cho thấy nó cung cấp sự cân bằng tốt hơn so với các thuật toán hiện có.

Ở bài toán thứ hai, chúng tôi tập trung vào việc giải quyết các thách thức trong mạng cảm biến không dây sạc lại, nơi một bộ sạc di động có thể được sử dụng để di chuyển và sạc các cảm biến. Bài toán quan trọng là thiết kế một chiến lược sạc hiệu quả cho bộ sạc di động trong khi phải đối mặt với các yếu tố bất định xảy ra trong mạng như nút bị lỗi hoặc nút mới được triển khai. Chúng tôi đề xuất một chiến lược sạc mới sử dụng phương pháp học tăng cường sâu để hướng dẫn bộ sạc di động một cách linh hoạt. Phương pháp của chúng tôi cho phép bộ sạc di động thích nghi với các thay đổi bất ngờ trong cấu trúc mạng nhờ các cơ chế mới trong học máy. Các thử nghiệm cho thấy tính ưu việt của mô hình của chúng tôi so với các phương pháp hiện tại trong việc kéo dài tuổi thọ mạng.

Các giải pháp đề xuất của chúng tôi cho hai vấn đề về vị trí nút trung gian và định tuyến bộ sạc di động có thể kéo dài tuổi thọ của mạng và giảm chi phí bảo trì đáng kể. Tiềm năng kết hợp hai kỹ thuật này trong mạng cảm biến sạc lại để tăng cường tính bền vững và đáng tin cậy của mạng là một hướng nghiên cứu tiềm năng trong tương lai. Việc áp dụng các giải pháp này, đặc biệt là những tiến bộ mới trong học tăng cường sâu, có thể mở đường cho các mô hình mạng cảm biến hiệu quả hơn, giúp chúng ta giám sát và quản lý tốt hơn các hệ thống khác nhau trong cuộc sống của chúng ta.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligent. viii

BS base station. viii, 19–22

DEM digital elevation model. viii, 21
DL deep learning. viii
DRL deep reinforcement learning. viii
DRL-TCC deep reinforcement learning approach for target coverage and connectivity problem. viii

EA evolutionary algorithm. viii

GA genetic algorithm. viii
GPrim Guided Prim NGSA-II. viii, 19, 22

IA intelligent agent. viii
INMA Invalid Node Minimized Algorithm. viii
IoT Internet of Things. viii

MC mobile charger. viii, 40, 41, 44–47, 49, 51
MDP Markov decision process. viii
MOEA multi-objective evolutionary algorithm. viii, 22, 38
MOEA/D multiobjective evolutionary algorithm based on decomposition. viii
MOO multi-objective optimization. viii, 30, 38
MST minimum spanning tree algorithm. viii

NEBP Node-Energy Bottleneck problem in multi-hop wireless sensor networks. viii, 19, 21, 22, 30, 38
NetKeys network random keys. viii
NJNP Nearest-Job-Next with Preemption. viii, 40
NSGA-ii non-dominated sorting genetic algorithm. viii

PF Pareto front. viii

QoS Quality of Service. viii

RL reinforcement learning. viii
RN relay node. viii, 20–22

SN sensor node. viii, 1, 13, 19–22, 31

WRSN wireless rechargeable sensor network. viii, 41
WSN wireless sensor network. viii, 13, 20, 21

List of Notations

B^{MC} battery capacity of the MC.
B^{SN} battery capacity of a sensor.
R a reward function.
T a transition model.
γ a discount factor (discount rate).
$\mathcal{A}$ a set of legal actions.
$\mathcal{M}$ Markov decision process.
$\mathcal{P}$ a set of deployed sensors.
$\mathcal{Q}$ a set of critical targets.
$\mathcal{S}$ a state space.
μ charging rate.
ν velocity of the MC.
ω_{move} ECR of the MC for traveling.
ω energy consumption rate.
π a policy.
τ charging trajectory.
$\tilde{E}_{td}$ energy requesting threshold.
a charging action.
e^{MC} residual energy of the MC.
e residual energy of a sensor.
m number of critical targets.
n number of deployed sensors.
p^{BS} base station.
p^{D} depot.
p a sensor.
q a critical target.
r_c communication range.
r_s sensing range.
x^D state of the depot.
x^{MC} state of the mobile charger.
x^{SN} state of a sensor.
x a state.

Chapter 1

Introduction

1.1 Motivation

In recent decades, the proliferation of electronic, communication, and computing technologies has fueled the rapid growth of the Internet of Things (IoT). Wireless Sensor Networks (WSNs), originally developed for military applications, are now one of the fundamental building blocks of the IoT, playing a crucial role in many civilian applications, including home automation (Pirbhulal et al., 2016), air/earthquake monitoring (Alphonsa and Ravi, 2016, Kingsy Grace and Manju, 2019), smart agriculture (Sanjeevi et al., 2020), health monitoring (Abdulkarem et al., 2020, Gardašević et al., 2020), smart cities (Csáji et al., 2017). A WSN typically consists of a few to hundreds or thousands of spatially dispersed and dedicated sensors to monitor specific targets or areas of interest. Each sensor node (SN) is equipped with sensing, processing, and communication capabilities to convert an analog signal of physical quantity into a digital signal and connect the node to the network. The sensing data will be cooperatively transmitted through the wireless network to a base station (BS), also known as a *sink*, where the data can be observed and analyzed.

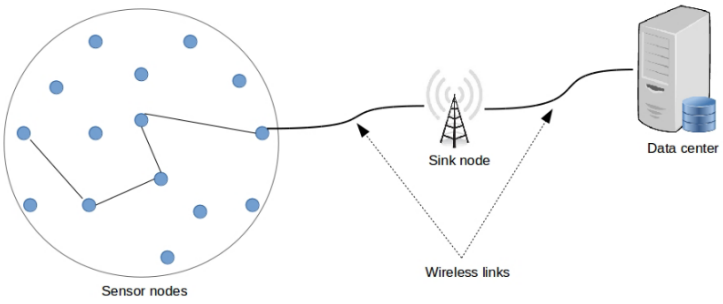


Figure 1.1.1: Wireless sensor network architecture (source: Bahri (2018)).

Such a design offers WSNs the capability of being deployed on the fly and can operate unattended, self-organizing without requiring any pre-existing infrastructure and with little maintenance. The sensor nodes collaborate to achieve a common goal, such as sensing and reporting temperature, humidity, or motion in a specific area. The communication among the sensor nodes is usually achieved through multi-hop routing, where the data is forwarded from node to node until it reaches the base station. Therefore, they can cover large areas and gather data from numerous sources simultaneously, providing real-time

monitoring of the target area. Additionally, the use of WSNs can reduce labor and maintenance costs since the SNs are equipped with low-cost, low-power batteries that can operate for extended periods (Akyildiz et al., 2002).

However, the limited energy capacity of the sensors poses a crucial challenge for ensuring continuous surveillance. Once the battery is fully consumed, the sensor can no longer monitor the targets or relay the data, leading to network fragmentation and data loss in some parts of the sensing field. Further, WSNs are often deployed in harsh and inaccessible environments for humans, such as underground tunnels and battlefields, making it challenging to replace the sensors' batteries. Hence, in most WSN applications, one of the primary objectives is to maximize the network's lifespan while keeping it functional to ensure continuous data transmission and monitoring of the targets (Yetgin et al., 2017).

The last two decades observed a remarkable effort of researchers put into designing new paradigms and protocols to prolong the network lifetime. The approaches can be broadly classified into two main categories: *sensor functioning optimization* and *energy replenishment*. Sensor functioning optimization aims to increase the efficiency of SNs and reduce energy consumption through methods such as energy-efficient routing (Raj et al., 2019), data aggregation (Goyal et al., 2019), and sensor scheduling (Haimour and Abu-Sharkh, 2019), while energy replenishment focuses on providing external sources of energy, such as energy harvesting (Adu-Manu et al., 2018, Shaikh and Zeadally, 2016) or wireless charging (Kaswan et al., 2022, Qureshi et al., 2022), to the sensor nodes.

One advantage of optimizing sensor functioning is that it can extend network lifespan without added hardware or infrastructure. These techniques can also save energy and be customized for various applications. However, they may not suffice for high-throughput needs that demand real-time data transmission, as this approach can only extend the sensors' lifetime for a certain amount of time. The battery will eventually be exhausted if no external source supplies the sensors. Meanwhile, energy replenishment provides a continuous energy supply to sensors, possibly eliminating the need for battery replacements. This technique works in remote or harsh environments but may require extra hardware or infrastructure, such as energy harvesters or wireless chargers. Hardware costs and maintenance are also potential drawbacks in some applications.

The choice of technique to prolong the network lifetime in wireless sensor networks depends on various factors, including application requirements, energy constraints, and cost considerations. In this thesis, we study two problems; each is an emerging technique that has attracted significant attention in recent years for its potential to prolong the network lifetime of WSNs. Specifically, we investigate (1) energy-efficient routing via relay node placement and (2) adaptive routing strategies for the mobile charger in wireless rechargeable sensor networks (WRSNs).

Energy-efficient routing via relay node placement. Relay node placement is an essential technique to optimize the functioning of WSNs. The idea is to deploy non-sensing relay nodes (RNs) to increase the network's capability and balance the energy consumption among nodes. As RNs are often determined after deploying SNs, we can optimize the placement of RNs to balance the energy consumption among nodes, which in turn prolongs the network lifetime. Past research efforts have focused on finding the minimum number of RNs required to reduce deployment costs while ensuring QoS criteria such as network coverage and connectivity. Recent works have also considered its potential to elongate the network lifetime. However, striking a balance between these criteria remains a significant challenge. To address this issue, we proposed a multi-objective approach

to constructing an efficient communication structure (routing tree) with the least possible number of RNs, thereby prolonging the network's lifetime while ensuring its connectivity.

Adaptive charging strategies in WRSNs. Despite remarkable progress in recent years, sensor function optimization and energy harvesting techniques cannot provide reliable service for high-throughput applications requiring continuous surveillance. Recent advancements in wireless charging provide a foundation for a novel scheme: wireless rechargeable sensor networks (WRSNs). The idea is to employ a mobile charger (MC) with a high-power battery to go around and charge the sensors wirelessly. The main challenge here is to design a suitable charging strategy for the mobile charger while accounting for the uncertainties arising in the network. Existing charging schemes either make a strict assumption about constant energy consumption rates or cannot adapt to unpredictable changes in the network topology. To overcome this challenge, we propose a novel charging scheme that uses a deep reinforcement learning (DRL) approach to guide the mobile charger adaptively. Our approach enables the mobile charger to adapt to spontaneous changes in the network topology.

1.2 Thesis contributions

The main contributions of this thesis can be summarized as follows:

- In this study, we examine common approaches for prolonging the network lifetime in wireless sensor networks, which can be classified into two main groups: sensor functioning optimization and energy replenishment. We investigate one emerging issue for each group, which can help in extending the network lifetime further.
- In Chapter 4, we investigate energy-efficient routing techniques by considering the relay node placement as a multi-objective problem. We introduce a novel multi-objective evolutionary algorithm that exploits problem-specific features to find a Pareto-front that balances the trade-off between the number of relay nodes and the network's energy consumption. We conduct extensive experiments to demonstrate the effectiveness of our method and show that it provides a better trade-off compared to existing algorithms. The paper summarizing the results is under review in the Soft Computing journal (Bui et al., 2023).
- In Chapter 5, we consider the on-demand charging problem in WRSN settings. We propose an adaptive charging scheme for the MC using deep reinforcement learning to choose the next charging destination. Our proposed framework is flexible as it can operate in a dynamic sensor network where the number of sensors might change due to node failures or deployments. The experiments show the superiority of our model compared to existing on-demand methods in prolonging the network lifetime. The results were published at the IEEE MASS, 2022 (Bui et al., 2022).

1.3 Thesis outline

The thesis is structured as follows:

In *Chapter 2*, we provide an overview of the evolutionary and deep reinforcement learning algorithms utilized in this study. We introduce the fundamental concepts and principles of these algorithms to provide readers with the necessary background knowledge for subsequent chapters.

In *Chapter 3*, we present a comprehensive survey of the network lifetime problem in WSNs. Our survey covers current state-of-the-art works in sensor functioning optimization and energy replenishment approaches. We also provide an in-depth analysis of energy-efficient routing approaches and adaptive charging schemes in WRSNs. This analysis will position our work better in the literature.

In *Chapter 4*, we delve into the relay node placement problem and present our multi-objective framework for addressing this problem. We describe the problem statement and the design of our algorithm, which takes into account problem-specific properties to find a Pareto-front that balances the number of relay nodes and the network's energy consumption. We also conduct experiments to demonstrate the effectiveness of our approach and compare it with existing algorithms.

In *Chapter 5*, we present our adaptive deep reinforcement learning framework for scheduling charging trajectories for mobile chargers in WRSN settings. We describe the design of our approach, which enables the mobile charger to choose the next charging destination adaptively, based on the current energy levels of the sensors and the network's topology. We demonstrate the effectiveness of our model through extensive experiments and compare it with existing on-demand charging methods.

Finally, in *Chapter 5.5*, we conclude the thesis by summarizing our contributions and highlighting the key findings of our study, and then providing recommendations for future works.

Chapter 2

Background

In this chapter, we present an overview of the evolutionary and deep reinforcement learning algorithms used in the study. The purpose of this chapter is to introduce readers to the basic concepts and principles of these algorithms, which will be necessary for understanding the subsequent chapters. We introduce multi-objective optimization (MOO) problems and describe how evolutionary algorithms simulate the process of natural selection to evolve optimal solutions to MOO problems in Section 2.1. We also explain how deep reinforcement learning algorithms combine neural networks and reinforcement learning to enable agents to learn from their experiences and improve their decision-making abilities over time in Section 2.2.

2.1 (Multi-objective) Evolutionary Algorithms

2.1.1 Multi-objective optimization problems

Multi-objective optimization (MOO) is a branch of optimization that deals with problems involving multiple conflicting objectives. In MOO, the goal is to find a set of solutions that simultaneously optimize multiple objectives. These objectives often conflict with each other, leading to no single solution being optimal for all objectives. Mathematically, a multi-objective optimization problem can be formulated as

$$\min_{\mathbf{x}\in\mathcal{X}} \ (f_1(x), f_2(x), \ldots, f_m(x)) \tag{2.1}$$

where $f_i : \mathcal{X} \to \mathbb{R}$ is the ith objective function. As there are conflicting criteria, we first need a definition of solution domination to compare between solutions.

Definition 2.1.1 (Solution domination)**.** *A solution $\mathbf{x}_1$ is said to dominate another solution $\mathbf{x}_2$, denoted $\mathbf{x}_1 \succ \mathbf{x}_2$, if $\mathbf{x}_1$ is no worse than $\mathbf{x}_2$ in all objectives and is strictly better in at least one objective.*

A solution $\mathbf{x}_1$ is said to be non-dominated if there is no other solution in the population that dominates $\mathbf{x}_1$. The Pareto front help represents the set of non-dominated solutions. A solution is Pareto optimal if there is no other solution that is better in all objectives. The Pareto front can be mathematically formulated as follows. Let $\mathbf{x} \in \mathcal{X}$ denote a solution vector, where $\mathcal{X}$ is the feasible region. Let $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})]$ denote the

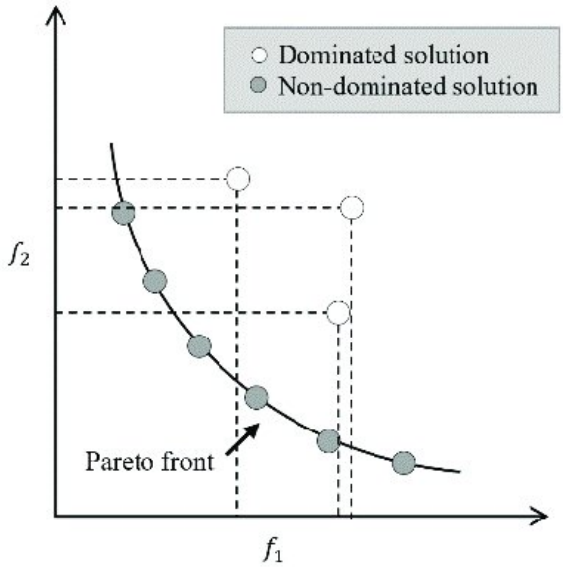


Figure 2.1.1: Pareto dominance (Source: Verma et al. (2021)).

vector of m objective functions. Then, the Pareto front can be defined as:

$$\mathcal{P} = \{\mathbf{x} \in \mathcal{X} | \nexists \mathbf{x}' \in \mathcal{X} \text{ such that } \mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})\},$$

where $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}')$ denotes that $\mathbf{x}'$ dominates $\mathbf{x}$, meaning that $\mathbf{x}'$ is no worse than $\mathbf{x}$ in all objectives and strictly better than $\mathbf{x}$ in at least one objective. Figure 2.1.1 shows an example of the Pareto front. The Pareto front is a useful tool for decision-making in MOO because it provides a set of trade-off solutions that decision-makers can choose from based on their preferences.

2.1.2 Non-dominated sorting genetic algorithm

Multi-objective evolutionary algorithms (MOEAs) are a class of optimization algorithms that are designed to tackle MOO problems. MOEAs are inspired by the process of natural selection, where individuals with better fitness are more likely to survive and reproduce. In MOEAs, candidate solutions are represented as individuals in a population, and the fitness of each individual is evaluated based on multiple objective functions. The goal of MOEAs is to find a set of Pareto optimal solutions.

Non-dominated Sorting Genetic Algorithm (NSGA) is one of the most popular MOO algorithms that uses the genetic algorithm (GA) as a search method. NSGA-ii was first introduced by Deb et al. (2002) as an improvement over the traditional GA for multi-objective optimization problems. The core idea behind NSGA is to sort the population of candidate solutions into several fronts based on their non-domination relationship. The first front contains non-dominated solutions, and the second front contains solutions that are dominated by solutions in the first front, and so on. By sorting the population into fronts, NSGA is able to maintain a diverse set of Pareto optimal solutions.

Similar to the original genetic algorithm, NSGA uses two main genetic operators: crossover and mutation. Crossover is a process of combining two parent solutions to generate a new offspring solution, while mutation is a process of introducing random changes to a solution. NSGA uses tournament selection to choose parent solutions for crossover and mutation. In each generation, NSGA performs non-dominated sorting to rank the

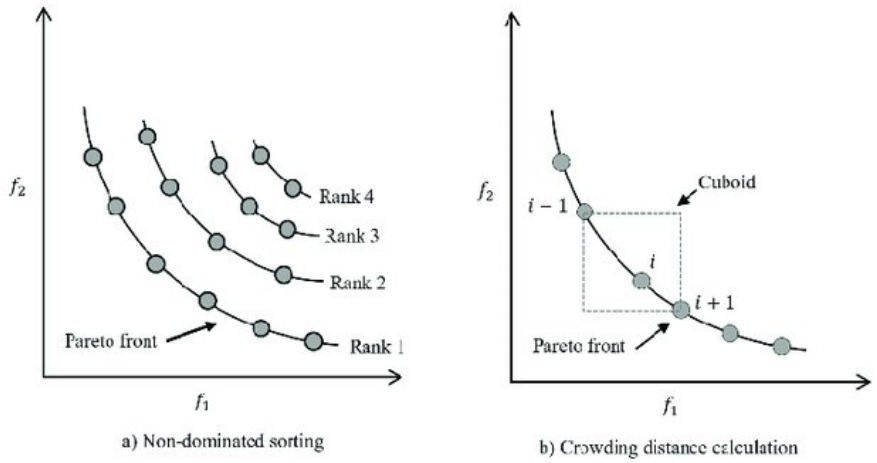


Figure 2.1.2: Examples of the non-dominated sorting algorithm and crowding distance calculation (Source Verma et al. (2021)).

population into fronts and assigns a crowding distance value to each solution in the front. The crowding distance value measures how crowded a solution is in the front based on the distance to its neighboring solutions. The crowding distance value of a solution $\mathbf{x}_i$ in front F_j is defined as:

$$\text{crowding\_distance}(\mathbf{x}_i, F_j) = \sum_{k=1}^{m} \frac{\mathbf{f}_k(\mathbf{x}_{i+1}, F_j) - \mathbf{f}_k(\mathbf{x}_{i-1}, F_j)}{\mathbf{f}_k^{max} - \mathbf{f}_k^{min}}, \tag{2.2}$$

where M is the number of objectives, $\mathbf{f}_k(\mathbf{x}_{i+1}, F_j)$ and $\mathbf{f}_k(\mathbf{x}_{i-1}, F_j)$ are the objective function values of the neighboring solutions of x_i in front F_j along the k-th objective, and $\mathbf{f}_k^{max}$ and $\mathbf{f}_k^{min}$ are the maximum and minimum objective function values, respectively, along the k-th objective over the entire population. The crowding distance value ensures that solutions that are diverse in the objective space are preserved in the population. NSGA then selects the best solutions from the fronts based on their rank and crowding distance and uses them to generate the next population. The selection operator ensures that solutions in the first front are always selected, and solutions in the subsequent fronts are selected based on their crowding distance value. We provide a pseudocode of NSGA-ii in Algorithm 1.

Algorithm 1 Non-dominated Sorting Genetic Algorithm

Input: Population size N, number of generations T, crossover probability p_c, mutation probability p_m, selection operator

Initialize population P_0 with N individuals
Evaluate objective functions for each individual in P_0
$t \leftarrow 0$
while $t < T$ **do**
  $Q_t \leftarrow$ Create empty population
  $R_t \leftarrow$ Create empty population
  $P'_t \leftarrow$ Perform tournament selection on P_t
  for $i \leftarrow 1$ to N by 2 **do**
    Select parents x_i, x_{i+1} from P'_t using binary tournament selection
    if Random number $< p_c$ **then**
      $y_i, y_{i+1} \leftarrow$ Perform crossover on x_i, x_{i+1}
    else
      $y_i, y_{i+1} \leftarrow x_i, x_{i+1}$
    end if
    for $j \leftarrow i, i+1$ **do**
      if Random number $< p_m$ **then**
        $y_j \leftarrow$ Perform mutation on y_j
      end if
      Evaluate objective functions for y_j
      $Q_t \leftarrow Q_t \cup y_j$
    end for
  end for
  Merge P_t and Q_t into R_t
  Perform non-dominated sorting on R_t to obtain fronts $F_1, F_2, \ldots, F_k$
  Set $P_{t+1} \leftarrow \emptyset$, $i \leftarrow 1$
  while $|P_{t+1}| + |F_i| \leq N$ **do**
    Perform crowding distance assignment on F_i
    $P_{t+1} \leftarrow P_{t+1} \cup F_i$
    $i \leftarrow i + 1$
  end while
  Sort remaining individuals in F_i by crowding distance
  Add top $(N - |P_{t+1}|)$ individuals in F_i to P_{t+1}
  $t \leftarrow t + 1$
end while
Return best individual(s) found in final population P_T

2.2 Deep Reinforcement Learning

2.2.1 Reinforcement learning and key concepts

Reinforcement learning (RL) is one of the fundamental paradigms of machine learning, alongside supervised learning and unsupervised learning. It involves an agent that interacts with an environment, receiving rewards for its actions. The objective of RL is to enable the agent to learn a policy that maximizes the rewards it receives by iteratively trying different strategies and receiving feedback. This way, the agent can adapt to changes in the environment and improve its performance over time. Figure 5.3.1 illustrates the

basic framework of reinforcement learning.

In RL problems, an agent interacts with an environment over a sequence of discrete time steps. At each time step, the agent observes a state of the environment, takes action, receives a reward, and transitions to a new state. The state of the environment at each time step depends only on the previous state and action and not on any states or actions that occurred before that.

RL problems are often represented as *Markov Decision Processes* (MDPs), which are characterized by a set of states $\mathcal{S}$, actions $\mathcal{A}$, transition probabilities T, reward function R, and a discount factor γ. The state space is the set of all possible states that the environment can be in. The action space is the set of all possible actions that the agent can take. The transition probabilities describe the probability of transitioning from one state to another when taking a particular action. The reward function maps each state-action pair to a scalar reward. The discount factor is a parameter that determines the relative importance of future rewards.

The goal of an RL agent in an MDP is to learn a policy π, which is a mapping from states to actions, that maximizes the expected cumulative reward over time. The optimal policy is the policy that maximizes the expected cumulative reward for all possible initial states. To maximize the cumulative rewards, the agent needs to determine an appropriate *policy* $\pi(s)$ that selects the best action in each state, as well as a *value function* $V(s)$ that estimates the future rewards that will be obtained by following the policy. The interaction between the agent and the environment involves a sequence of actions that cause the environment to change state. This sequence can be described as an episode that ends when the environment reaches a terminal state. At each time step $t \in 1, 2, ..., T$, the agent observes the current state X_t, takes action A_t, and receives a reward R_t. The trajectory is a sequence of random variables:

$$\tau = \{X_0, A_0, \ldots, X_{T-1}, A_{T-1}, X_T\}.$$

The model gives all the necessary information about an environment: transition probability function T and reward function R. At state x_t, the agent chooses an action a_t, which leads to a new state x_{t+1} and receives a reward r_{t+1}. This is a *transition* step $(x_t, a_t, x_{t+1}, r_{t+1})$. The probability of this transition is:

$$\Pr(x_{t+1}, r_{t+1}|x_t, a_t) = \mathbb{P}[X_{t+1} = x_{t+1}, R_{t+1} = r_{t+1}|X_t = x_t, A_t = a_t]. \tag{2.3}$$

The policy π models the agent's behavior at a state x_t $\pi(a_t|x_t) = \mathbb{P}_\pi[A_t = a_t|X_t = x_t]$ and the *return* G_t is a accumulated discount rewards: $G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The discounting factor $\gamma \in [0, 1]$ determines how much the agent cares about rewards in the distant future relative to those in the immediate future. The value function is the expected return of state s at time t, which can be calculated by:

$$V_\pi(x_t) = \mathbb{E}_\pi[G_t|X_t = x_t]. \tag{2.4}$$

Similarly, the Q-function measures the quality of a state-action pair:

$$Q_\pi(x_t, a_t) = \mathbb{E}_\pi[G_t|X_t = x_t, A_t = a_t]. \tag{2.5}$$

The following equation draws the connection between the value function and the Q-function.

$$V_\pi(s_t) = \sum_{a_t \in \mathcal{A}} Q_\pi(x_t, a_t)\pi(a_t|x_t). \tag{2.6}$$

In some cases, it is favorable to use the difference in return of a state-action pair compared to the expected return of that state. We define that difference as the *advantage value*:

$$A_\pi(x_t, a_t) = Q_\pi(x_t, a_t) - V_\pi(x_t). \tag{2.7}$$

Optimal value and policy. The optimal value function produces the maximum return:

$$V^\star(x) = \max_\pi V_\pi(x), \quad Q^\star(x, a) = \max_\pi Q_\pi(x, a). \tag{2.8}$$

The optimal policy achieves optimal value functions:

$$\pi^\star = \arg\max_\pi V_\pi(x) = \arg\max_\pi Q_\pi(x, a). \tag{2.9}$$

Thus, $V_{\pi^\star}(x) = V^\star(x)$ and $Q_{\pi^\star}(x, a) = Q^\star(x, a)$.

Bellman equations. The Bellman equation expresses the relationship between the value of a state and the values of its successor states. It states that the value of a state is equal to the immediate reward obtained by taking action in that state plus the discounted value of the next state that the agent transitions to.

$$V(x_t) = \mathbb{E}[R_{t+1} + \gamma V(X_{t+1})|X_t = x_t], \tag{2.10}$$

$$Q(x_t, a_t) = \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} Q(X_{t+1}, a_t)|X_t = x_t, A_t = a_t]. \tag{2.11}$$

Traditional RL methods use a combination of model-free techniques (e.g., Q-learning, SARSA) and model-based approaches (e.g., dynamic programming) to learn optimal policies in environments with discrete and small state-action spaces.

Deep reinforcement learning (DRL) is a recent extension of RL that leverages deep neural networks to enable learning in the high-dimensional state and action spaces. In DRL, the agent learns to directly map raw sensory inputs (e.g., pixel values in images) to actions without relying on hand-engineered features. This is achieved by combining deep neural networks with traditional RL algorithms, such as Q-learning and policy gradient methods. As this thesis uses actor-critic algorithms, which are highly based on policy gradient methods, we provide background knowledge about policy gradient methods in the following section.

2.2.2 Policy Gradient methods

Policy gradient methods are a class of reinforcement learning algorithms that aim to optimize policies directly rather than computing a value function and then deriving a policy from it. These methods have gained popularity recently due to their ability to handle continuous action spaces and their success in achieving state-of-the-art results in various applications, including game-playing, robotics, and natural language processing.

Policy gradient methods learn the policy directly with a parameterized function with respect to θ, $\pi(a|x;\theta)$. We train the agent to maximize the expected return. Specifically,

in continuous space with X_1 as the initial starting state:

$$J(\theta) = V_{\pi_\theta}(X_1) = \mathbb{E}_{\pi_\theta}[G_1|X_1]. \tag{2.12}$$

We maximize the objective function $J(\theta)$ using the gradient ascent method. The gradient of J can be computed using Policy Gradient Theorem Sutton and Barto (2018) as follows.

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|x;\theta) Q_\pi(x,a)]. \tag{2.13}$$

Having the gradient, we can optimize policies directly using gradient ascent algorithms. However, early policy gradient methods suffered from high variance, which made them difficult to converge to optimal policies.

To address this issue, actor-critic methods combine a policy network (the actor) with a value function network (the critic) to provide a more stable estimate of the policy gradient. Specifically, two networks are maintained in actor-critic methods. One network is used for learning value function, namely the *critic*, denoted V_ψ, and another learns the mapping between state and actions directly, known as the *actor*, denoted π_θ. The critic is used to criticize the actions made by the actor, and the actor adjusts its parameters in the direction suggested by the critic. The gradient of the actor is now given by:

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|x,\theta)(G_t - V_\psi(x))], \tag{2.14}$$

where $V_\psi(s)$ is the estimated value of state s given by the critic. This algorithm is called REINFORCE with a baseline.

2.2.3 Attention mechanisms

Over the last decade, encoder-decoder architecture has emerged as one of the most prominent deep learning architectures. Originally introduced to solve the problem of mapping fixed-length input to output in sequence-to-sequence learning, the vanilla encoder-decoder encodes a variable-input sequence to an internal, fixed-dimensional representation. This representation is then used by an RNN-based decoder to produce a variable-length output until a termination criterion is detected. However, a major drawback of this approach is its inability to remember long sentences. To overcome this limitation, attention mechanisms were introduced. Attention allows the decoder to use any of the encoder's hidden states instead of relying on the fixed-length representation produced by the encoder. This is achieved by creating shortcuts that combine the entire input sequence into a context vector, with weights assigned to represent how much attention is devoted to each input. Mathematically, let us denote the encoder and decoder hidden states with $(e_1, e_2, ..., e_n)$ and $(d_1, d_2, ..., d_n)$ a context vector at decoding time i is given by:

$$c_i = \sum_{j=1}^{n} a_j^i e_j, \tag{2.15}$$

where a^i is an *alignment* of the input vector, which is calculated by:

$$a_j^i = \text{softmax}(u_j^i), \quad j \in \{1, 2, ..., n\}, \tag{2.16}$$

where:

$$u_j^i = f(W_1 e_j + W_2 d_i), \quad j \in \{1, 2, ..., n\}. \tag{2.17}$$

This context vector c is later concatenated with decoder state d to make a prediction or compute a hidden vector for the next steps of the recurrent model.

2.2.4 Pointing mechanism

The pointing mechanism is a technique first proposed in (Vinyals et al., 2015) to produce discrete outputs that correspond to positions in the input. For example, in the combinatorial problem - Travel Sailing Problem, the solution is a permutation of the input positions. In (Vinyals et al., 2015), the authors proposed a Pointer network which is an encoder-decoder LSTM model. The input, including a sequence of the node's position, is encoded by an LSTM encoder. In the decoder, instead of blending the encoder hidden states e_j into a context vector c at each decoder step, a reduction of attention mechanism is used to point to a member of the input sequence to be selected as the output:

$$u_j^i = f(W_1 e_j + W_2 d_i), \quad j \in \{1, 2, ..., n\}, \tag{2.18}$$

$$a_j^i = \text{softmax}(u^i), \quad j \in \{1, 2, ..., n\}, \tag{2.19}$$

where a_j^i is considered as the probability to select input j in the decoder step i. The node with the highest probability is chosen to be visited next. The procedure is iteratively repeated to obtain the final solution.

Chapter 3

Literature Review

3.1 Network lifetime in WSNs

A wireless sensor network (WSN) is comprised of several low-cost, low-power SNs, ranging from a few to thousands in number, that connect with each other through a wireless network. Each SN is composed of four primary components: (1) a *sensor unit* that converts the physical quantity's analog signal into a digital signal, (2) a *preprocessing unit* that enables computational and storage capabilities, (3) a *transceiver unit* that connects the node to the network, and (4) a *power unit*, typically an electrochemical battery (Akyildiz et al., 2002). Figure 3.1.1 provides a simplified diagram of the sensor node architecture.

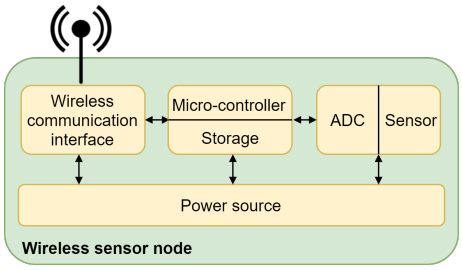


Figure 3.1.1: A block diagram of the architecture of the sensor node in the WSN.

Sensor nodes are generally low-cost, resulting in a battery with a low capacity and usually non-renewable. Therefore, extending the network's lifetime is a crucial factor that determines the overall efficiency and effectiveness of these networks. The network lifetime is generally defined as the time until either coverage or connectivity is lost (Zhao and Gurusamy, 2008). Numerous efforts have been dedicated to extending the lifespan of wireless sensor networks (WSNs), which can be classified into two main groups: *sensor functioning optimization* and *energy replenishment*.

Sensor functioning optimization focuses on enhancing the efficiency of sensor nodes while reducing their energy consumption. Various methods have been proposed in this area, such as data reduction, which involves removing redundant or unnecessary data to reduce the amount of information transmitted and, consequently, the energy consumption (Goyal et al., 2019). Another approach is sleep/wakeup schemes, which enable sensors to alternate between sleep and active modes based on predefined schedules or events, thereby reducing energy consumption during idle periods (Haimour and Abu-Sharkh, 2019). Additionally, energy-efficient routing protocols have been developed to minimize energy

consumption by reducing the number of hops between nodes and selecting paths with low energy costs (Raj et al., 2019).

These above methods have the advantage of being able to prolong the network lifetime without requiring additional hardware or infrastructure and can be implemented at the software level. They also have the potential to achieve significant energy savings and can be tailored to suit different application requirements. However, these techniques may not be sufficient for high-throughput applications that require real-time or continuous data transmission, as this approach can only extend the sensors' lifetime for a certain amount of time. The battery will eventually be exhausted if no external source supplies the sensors.

On the other hand, energy replenishment aims to provide external sources of energy to the sensor nodes to extend their lifetime. One promising approach is energy harvesting, which involves converting ambient energy from the environment, such as solar, thermal, or kinetic energy, into electrical energy to power the sensors (Adu-Manu et al., 2018). Nevertheless, this technique dramatically depends on an ambient source that is usually unstable and uncontrollable. Another approach is wireless charging, which involves wirelessly transmitting energy to the sensors using electromagnetic waves or magnetic resonance. The idea is to employ a (or multi-) mobile charger (MC), which is equipped with a high-capacity battery and a transmission coil, to travel around the sensing field and charge the sensors wirelessly (Qureshi et al., 2022).

Energy replenishment techniques can provide a continuous supply of energy to the sensor nodes without the need for battery replacements. They also have the advantage of being able to operate in remote or harsh environments where traditional power sources may not be available. However, these techniques may require additional hardware and infrastructure, such as energy harvesters or wireless chargers. Moreover, the cost and maintenance of such hardware and infrastructure may be a concern in some applications.

In the following sections, we focus on two problems of energy-efficient routing via relay node placement and charging policies for WRSNs.

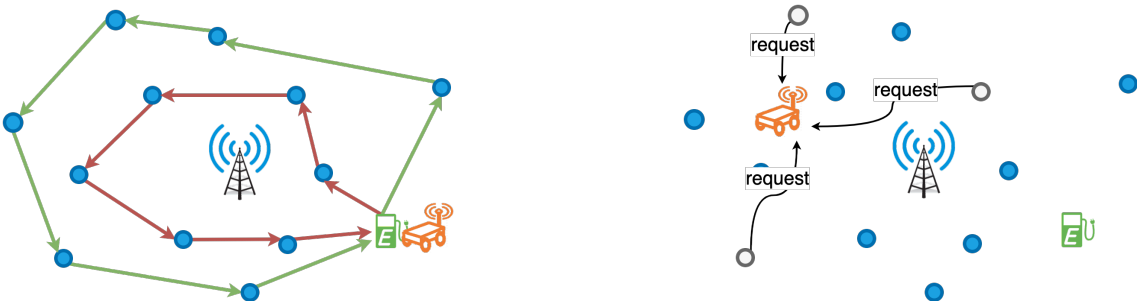
3.2 Energy-efficient routing via relay node placement

Energy-efficient routing is a well-established research area that aims to reduce energy consumption while maintaining reliable data delivery (Behera et al., 2022, Raj et al., 2019). Traditional approaches mostly focused on the existing structure of the WSNs to determine the routing protocols. Popular ones include hierarchical routing protocols, data-centric routing protocols, and location-based routing protocols. Hierarchical routing protocols like LEACH(Heinzelman et al., 2000) divide the network into clusters to minimize energy consumption, while data-centric routing protocols like Directed Diffusion (Intanagonwiwat et al., 2003) route data based on the content of the message. Location-based routing protocols like GRP (Karp and Kung, 2000) use location information to make routing decisions. However, the effectiveness of these traditional routing protocols can be limited by the existing network topology and node density.

In recent years, there has been a growing interest in relay node placement strategies to enhance energy consumption in wireless sensor networks (Verma et al., 2015). The idea is to deploy additional non-sensing relay nodes to increase the network's capability and balance the energy consumption among nodes. Originally, relay node placement is designed to enhance to QoS criteria of WSNs, such as network connectivity and fault tolerance (Hanh et al., 2019, Lee et al., 2015, Ma et al., 2015, Sheikhi et al., 2021). Re-

cently, relay node placement has been considered more for its potential to help balance the energy consumption among nodes, which in turn elongates the network lifetime (Tam et al., 2020). Tam et al. (2020) have considered two objectives: minimizing the number of relay nodes and minimizing the maximum node energy consumption to prolong the network lifetime. They proposed a weighted-sum approach to finding a routing tree that maximizes the network's lifetime with minimum additional relay nodes. Although their work is limited to 2-hop WSNs and employs only the weighted-sum algorithm, it shows the potential of applying relay node replacement in prolonging the network lifetime. This thesis aims the enhance the existing results of Tam et al. (2020) to multi-hop networks with a novel objective-oriented multi-objective algorithm. We will discuss this problem in detail in Chapter 4.

3.3 Charging policies in WRSNs



(a) Offline charging scheme

(b) Online charging scheme

Figure 3.3.1: A comparison of offline and online charging scheme.

Wireless Rechargeable Sensor Networks (WRSNs) are a subset of WSNs that use wireless charging technology to replenish the sensor nodes' energy. This technology provides a promising solution to one of the most significant challenges facing WSNs, which is the limited energy capacity of the sensors. WRSNs consist of a mobile charger that traverses the sensing field and wirelessly charges the sensors that need energy. This approach ensures continuous operation of the WSN without requiring manual battery replacements, making it ideal for remote or harsh environments where accessing the sensor nodes is difficult. However, constructing an efficient charging policy for mobile chargers (MCs) to meet the dynamic charging requirements of the sensors is one of the most critical challenges in WRSNs. Various proposals have been put forward to address this challenge, and they can be broadly classified into two main categories: offline charging schemes and online charging schemes (Figure 3.3.1).

Offline charging scheme. Offline charging schemes aim to optimize the charging policy before the charging process begins, usually by considering the expected workload, the capacity of the MCs, as well as the energy demand of the sensors. This approach enables the MC to charge the sensors with the optimal charging schedule, minimizing the overall energy consumption and extending the network lifetime. Lyu et al. (2019) propose a periodic charging planning for mobile Wireless Charging Equipment with limited traveling energy. They propose a Hybrid Particle Swarm Optimization Genetic Algorithm (HPSOGA) because of the NP-hardness of the problem. In (Jiang et al., 2017), the authors jointly consider charging tour planning and MC depot positioning for large-scale WSNs. Their method consists of charging tour planning, candidate depot identification

and reduction, depot deployment, and charging tour assignment. The charging scheme also considers the association between the MC charging cycle and the sensor nodes' lifetime. Ma et al. (2018b) aim to minimize the sensor energy expiration time and the charging tour length of the mobile charger. They develop an approximation algorithm for the charging utility maximization problem if the energy consumption of the mobile charger on its charging tour is negligible and an efficient heuristic through a non-trivial reduction from a length-constrained utility maximization problem otherwise. However, this approach requires a strict assumption about the constant energy consumption rate of sensor nodes, which is unrealistic in practice.

Online charging scheme. Online charging schemes determine the optimal charging policy while the charging process is ongoing, usually by monitoring the energy levels of the sensors and the remaining energy of the MC. Online charging schemes are generally more flexible and adaptive, as they can adjust the charging policy in response to changing network conditions. In the on-demand charging problem, sensor nodes request charging from the MC when their energy is depleted or falls under a predefined threshold. The MC maintains a pool of these requests and determines the next sensor to charge among the requested ones in the pool. The NJNP (He et al., 2013) algorithm charges the closest sensor node in the queue, while DWDP (Lin et al., 2019) uses double warning thresholds and double preemption to optimize charging priorities and recharge deadlines. ESync (Fu et al., 2015) constructs nested TSP tours to reduce travel distance and charging delay, and TSCA (Lin et al., 2017) minimizes the number of failed nodes while maximizing energy efficiency. Kaswan et al. (2018) present a Linear Programming (LP) formulation for the on-demand scheduling problem and then introduce an efficient solution based on a gravitational search algorithm (GSA) to tackle the problem. PA and INMA (Zhu et al., 2018) are two efficient online charging algorithms that first consider dynamic energy consumption rates based on their history statistics and real-time energy consumption. Recently, reinforcement learning-based algorithms (Cao et al., 2021, La et al., 2020) have also been considered for designing on-demand charging schemes. However, a common drawback of the above on-demand algorithms is the dependence on the chosen threshold for the charging requests, making it sensitive to that setting. We propose in this thesis a novel adaptive charging scheme by eliminating the on-demand charging request and using deep reinforcement learning to train the policy. The detail will be discussed in Chapter 5.

Chapter 4

An Evolutionary Algorithm for Optimal Node Placement

This chapter investigates the energy-efficient routing problem in WSNs through relay node placement. Existing approaches solely focus on minimizing the number of used relay nodes without considering energy consumption among nodes. We propose a relay node placement approach in multi-hop wireless sensor networks with two objectives: minimizing the number of used relay nodes and minimizing the maximum node energy consumption. The first objective is to restrict the deployment costs, while the second is to balance the energy consumption among nodes, which in turn extends the network's lifetime. To improve the network's reliability, we also consider a hop count bound, which acts as a delay constraint for transmitting packages. To solve our problem, we propose a novel objective-oriented multi-objective evolutionary algorithm (MOEAs) that leverages the problem-specific properties to improve the algorithm's convergence rate. Simulation results on 3D datasets show that our algorithm outperforms existing algorithms in all measured metrics.

4.1 Introduction

Relay node placement with a hop count bound is an emerging technique to enhance connectivity, lifetime, and reliability in multi-hop wireless sensor networks (Farsi et al., 2019, Lin et al., 2020, Priyadarshi et al., 2020, Verma et al., 2015). The key idea is to deploy non-sensing RNs to increase the network's capability and balance the energy consumption among nodes, enhancing the connectivity, lifetime, and fault tolerance of the WSN. Besides, deploying additional RNs could also shorten a long-hop transmission, which is much more expensive than multiple short-hops (Haenggi and Puccinelli, 2005).

However, optimizing the placement of RNs is a challenging task as it often involves multiple conflicting criteria. There are two variants of relay node placement problems in the literature: unconstrained and constrained problems. In the former problem, relay nodes can be placed anywhere in the terrain. In contrast, to avoid unrealistic relay deployments due to physical constraints, the latter restricts the position of the additional relay nodes at certain locations, which are determined in advance. However, both typically form NP-hard problems (Lloyd and Xue, 2006, Misra et al., 2009); thus, in practical settings, it is hard to obtain an optimal solution in a suitable amount of time.

Numerous works have been carried out to approximate the placement of RNs while

ensuring several constraints of connectivity and fault tolerance. Ma et al. (2015) studied the constrained relay node placement in WSNs and proposed a connectivity-aware local search algorithm to find the minimum number of relay nodes so each sensor is covered by at least one relay node. Lee et al. (2015) assured the fault tolerance of a partitioned WSN by establishing a bi-connected inter-partition topology while still deploying the least count of relay nodes. Bagaa et al. (2017) leveraged a Rayleigh block-fading channel and weighted communication graph to construct a routing tree with a minimum number of additional relay nodes. Hanh et al. (2019) introduced a multi-objective problem that simultaneously considers the target coverage, connectivity, and fault tolerance of WSNs. Sheikhi et al. (2021) proposed the two phases approach to provide multi-path routing and fault-tolerance with higher network connectivity in heterogeneous WSNs.

Most of the above works consider multi-hop communication to ensure connectivity and prevent long-hop transmission. However, unlimited hop communication could increase the network's latency and reduce its reliability Bhattacharya and Kumar (2014). As it is difficult to measure network latency before node deployments, a hop count bound is often used as a surrogate constraint for network latency (Bhattacharya and Kumar, 2014, Liang et al., 2019, Ma et al., 2017, 2018a). Bhattacharya and Kumar (2014) first studied a constrained relay placement problem with hop count bound and showed its NP-hardness. They then proposed a polynomial time approximation algorithm for the problem. Ma et al. (2017, 2018a) also used the hop count to measure delay and reliability and formulated the 2-connected hop-constrained relay node placement (HCRNP) problem. Two approximation algorithms are proposed to solve this problem. Liang et al. (2019) later conducted extensive real-world deployments of WSNs using existing algorithms and then proposed a Set-Covering-based Algorithm (SCA) to ensure the quality of communication in the network with a hop count bound as a delay constraint.

Despite the promising results, a drawback of the aforementioned works is the lack of considering the energy consumption of nodes in the placement. The energy consumption of the nodes in a WSN is well-known to be imbalanced since it depends heavily on the number of relayed packets and the distance to the next node in the network topology (Guleria and Verma, 2019). Thus, balancing the load among nodes is essential to prolong the network's lifetime. However, the network's lifetime and the cost of deploying additional relay nodes are two conflicting criteria. Deploying more relay nodes typically increases the network's capability and provides more possibilities for load balancing but induces more cost in the deployment. Recently, Tam et al. (2020) first considered these two objectives in their design. They proposed a weighted-sum approach to finding a routing tree that maximizes the network lifetime with minimum additional relay nodes. However, they only consider the 2-hop WSNs, which are only suitable for small networks. Additionally, the weighted-sum strategy must make certain assumptions when assigning weight values regarding how 'important' a criterion is compared to the other.

To overcome these issues, we introduce a novel problem, called Node-Energy Bottleneck Problem (NEBP), which considers both objectives: minimize the number of relays used and maximize the network lifetime in a multi-hop setting with a hop count bound. This paper aims to establish a communication structure (routing tree) that has balance communication among nodes with minimal additional RNs. The main difference compared to Tam et al. (2020) is that we consider a multi-hop scheme with a delay constraint by limiting the maximum number of communication hops for each SN towards BS. Moreover, we focus instead on using MOEAs to solve two objectives simultaneously.

MOEAs are favorable for their ability to provide Pareto fronts of non-dominated solu-

tions in the objective function space. These Pareto fronts endow decision-makers to select a solution that fits them best. In evolutionary-based approaches, a population of candidate solutions is maintained and evolved toward better solutions. There are two main types of representation of an individual in the population: *indirect* and *direct*.

In an indirect representation, the candidate solutions are mapped to a different space where standard crossover and mutation operators can be applied. As we are focusing on constructing a routing tree, a standard approach could use Prufer encoding (Prüfer, 1918), link and node biased (Palmer and Kershenbaum, 1994), or Network random keys (NetKeys) (Rothlauf et al., 2002) as the solution encoder. Recently, Prakash et al. (2020) leveraged a permutation encoding with a heuristic decoder to propose a hybrid multi-objective evolutionary algorithm (HMOEA) to find a minimal spanning tree with a minimum diameter. The advantage of indirect representation is that we can adopt standard operators directly on the solution representaiton (Nayyar et al., 2018). However, most of these representations suffer from the low locality (small changes in the code can lead to large changes in the decoded tree) (Prüfer, 1918), or infeasible and redundant representations (Prakash et al., 2020, Rothlauf et al., 2002).

On the other hand, direct representations can use a simple encoding method such as edge sets encoding (Raidl and Julstrom, 2003) and then perform a problem-specific crossover and mutation directly on phenotypes to create new offspring (Rothlauf, 2006). The main advantage of this scheme is the ability to apply a heuristic to guide search operators Hao and Liu (2017). Therefore, in this paper, we use edge sets encoding to represent the solutions and then propose the novel crossover and mutation operators to solve the Node-Energy Bottleneck problem in multi-hop wireless sensor networks (NEBP).

We outline the contributions of this chapter as follows.

- First, we introduce a novel problem called the Node-Energy Bottleneck (NEBP), which considers multi-hop networks with a hop count bound. We aim to minimize two objectives: i) the number of used relay nodes; ii) the maximum node energy consumption to prolong the network lifetime.
- Secondly, we propose Guided Prim NGSA-II (GPrim) to solve the proposed problem. The novelties of the proposed GPrim can be summarized as follows: i) according to the problem-specific characteristics, encoding-based edge-set and decoding methodologies are developed to represent the solution space; ii) we leverage the problem's energy property to develop a heuristic Prim-based crossover and two mutations including energy-oriented mutation and relay-oriented mutation to improve the convergence rate of the algorithm.
- The proposed algorithm is validated against different encoding methods, including Permutation, Prufer code, NetKeys, and Edge sets. The comparison is delivered on various metrics showing that our algorithm outperforms existing approaches by a significant margin.

4.2 Problem statement

4.2.1 Network structure

We consider the deployment of a wireless underground sensor network as in Tam et al. (2020), with a multi-hop network structure as described in Wu and Liu (2013). A network includes a base station (BS), a set of SNs, and a set of SNs deployed in three-dimensional

terrains. We consider two types of connection: relay nodes - base station and sensor nodes - sensor nodes/relay nodes. Sensing data is gathered by SNs and sent to a BS through a relay node or other sensors. The data transmitted to relay nodes can only be forwarded directly to the base station rather than other sensors or relays. We assume the sensor network is static, meaning SNs have already been deployed, and a finite set of potential positions for SNs is known in advance. The base station is a sink node deployed at the central terrain with an unlimited power supply while relay nodes (RNs) and SNs have the same initial energy, which cannot be replenished.

4.2.2 Energy consumption model

Numerous energy dissipation models in WSNs are studied with different assumptions. In this work, we use the same energy model as in Gawade and Nalbalwar (2016), which accounts for the dissipated energy at both the receiver and transmitter during a transmission. The free space model (d^2 power loss) is used for proximal transmissions, and the multi-path fading model (d^4 power loss) is considered for large-distance transmissions. Thus the energy dissipated by the transmitter for transmitting an l-bit packet to a distance d is given by:

$$\tilde{E}_t(d) = \begin{cases} l\epsilon_{elec} + l\epsilon_{fs}d^2 & \text{if } d \leq d_0 \leq r_c, \\ l\epsilon_{elec} + l\epsilon_{mp}d^4 & \text{if } d_0 < d \leq r_c, \\ \infty & \text{if } r_c < d, \end{cases} \tag{4.1}$$

where $d_0 = \sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}}$ is the distance threshold for swapping amplification models and r_c indicates the range with which a node can communicate. In other words, no connection will be established among the nodes out of this range.

The energy consumption of the receiver to receive an l-bit packet is calculated as follows:

$$\tilde{E}_r = l\epsilon_{elec}. \tag{4.2}$$

The dissipated energy of a node receiving η packets and transmitting them to the parent node is calculated by the following formula

$$\tilde{E}(\eta, \zeta, d) = \eta\tilde{E}_r + (\eta + \zeta)\tilde{E}_t(d), \tag{4.3}$$

where $\tilde{E}_t(d), \tilde{E}_r$ are calculated as in Equation 4.1 and 4.2, respectively. The argument ζ equals 1 if the node is a sensor node, and 0 otherwise. The argument d is the transmission distance. The network parameters shown in Table 4.2.1 are set as in Wu and Liu (2013).

Table 4.2.1: Network parameters.

Parameter	Value
ϵ_{elec}	$50nJ/bit$
ϵ_{fs}	$10pJ/bit/m^2$
ϵ_{mp}	$0.0013pJ/bit/m^4$
ϵ_{DA}	$5pJ/bit$

4.2.3 Problem formulation

We consider a wireless sensor network including a set of deployed sensor node $S = \{s_1, s_2, ..., s_{n_s}\}$, a set of potential relay nodes $R = \{r_1, r_2, ..., r_{n_r}\}$ and a base station denoted as s_0. The position of each node is represented as a single point in a 3D space that is interpolated from the digital elevation model (DEM) model (Florinsky, 2016)). The communication between two nodes can only be established if the Euclidean distance between them does not exceed the communication range r_c.

We want to find a routing tree that balances the energy consumption among nodes with a minimal number of deployed RNs. Here, we consider the max-hop constraint $\hbar$ that limits the maximum number of communication hops for each SN towards BS (depth of the routing tree). We denote the problem as the Node Energy Bottleneck problem with hop count bound. The formal formulation below models the desired structure as a Steiner tree (Hwang and Richards, 1992).

Definition 4.2.1 (Steiner tree)**.** *Given an undirected graph $G = (V, E)$ and a set of terminal nodes $N \subseteq V$. A tree $T = (V_T, E_T)$ is called a Steiner tree if it contains no cycles and spans all terminal nodes, $N \subseteq V_T \subseteq V$. The set of nodes $V_T \setminus N$ is called Steiner nodes.*

Input:

- $G = (V, E)$ is an undirected graph, where $V = S \cup R \cup \{s_0\}$ is set of vertices in the graph, S is set of sensor nodes, R is set of relays, and s_0 corresponds to base station.
- $N = S \cup \{s_0\}$ denotes the set of terminal nodes.
- $r_c \in \mathbb{R}^+$ is the communication range.
- $d : V \times V \to \mathbb{R}^+$ is the distance function. An edge $e = (u, v) \in E$ only if $d(u, v) \leq r_0$.
- $\hbar \in \mathbb{N}^+$: the max-hop constraint.

Constraints:

- Every selected RNs (Steiner points) directly connect to base station s_0 (denoted as node 0)
$$(s_0, v) \in E_T \quad \forall v \in V_T \cap R.$$
- The unique path from a specified root s_0 to any other node has no more than $\hbar$ hops (edges)
$$\text{len}(s_0, v) \leq \hbar \quad \forall v \in V_T,$$
where $\text{len}(u, v)$ denotes the length of the unique path between two nodes u and v.

Output: A valid solution is a Steiner tree $T = (V_T, E_T)$ that spans the set of terminal nodes $N = S \cup \{s_0\}$ and satisfies the above constraints.

Figure 4.2.1 shows an example of a network with three relays and six sensors.

Objectives: The Node-Energy Bottleneck problem in multi-hop wireless sensor networks (NEBP) seeks a Steiner tree $T = (V_T, E_T)$ in the valid output space that optimizes two following objectives:

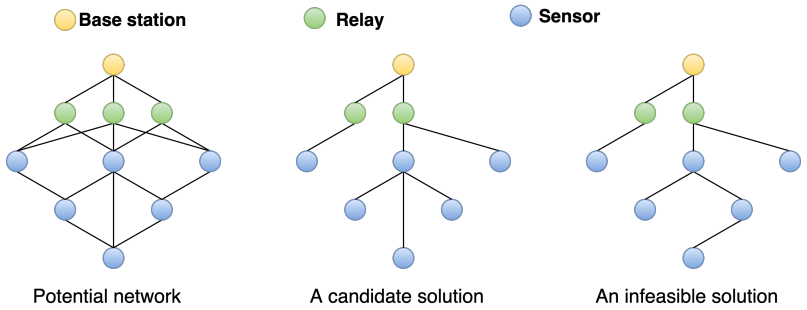


Figure 4.2.1: An example of a network with 3 relays, 6 sensors, and max-hop constraint is 3.

- Minimize the number of selected SNs (Steiner node):

$$|V_T \setminus N| \to \min . \tag{4.4}$$

- Minimize the maximum energy consumption of each node:

$$\max_{v \in V_T \setminus \{s_0\}} \tilde{E}_v(\eta, \zeta_v, d) \to \min, \tag{4.5}$$

  where $\tilde{E}_v$ is calculated as Equation 4.3.

4.3 Proposed method

We propose a phenotype-based multi-objective evolutionary algorithm (MOEA) named *Guided Prim NGSA-II (GPrim)* to solve the NEBP. In this scheme, the population is first initialized by a random-tree algorithm in which the candidate solutions are encoded by the edge sets encoding method. Then, we apply the NSGA-II algorithm (Deb et al., 2002) to maintain and evolve the population towards better solutions using the problem-specific search operators. We leverage the problem's energy property to develop a heuristic Prim-based crossover and two objective-oriented mutations to reduce ineffective moves from standard search operators. The specifics of solution representation, initialization, crossover, and mutation are described below.

4.3.1 Solution representation

Although a direct representation needs no mapping between the phenotypic and genotypic space, a data structure is still necessary for processing Li (2001), Rothlauf (2006). We use edge-set encoding on this problem for its simplicity. This encoding can act as the basis for evaluating the solution or be converted to an adjacency list in linear time. As we want to find a Steiner Tree that can connect all SNs to the BS, the number of vertices in the solutions is not consistent. For simplicity, we initialize solutions with the connections from the BS to every RN, and this structure is maintained in all candidate solutions in the population. The RNs with no connection to any SNs are later removed from the output structure by the decoder.

4.3.2 Initialization

In the hop-constrained spanning tree problem, the number of hops in a rooted tree (h) is bounded by its diameter (d):

$$d/2 \leq h \leq d.$$

Therefore, we leverage PrimRST (Raidl and Julstrom, 2003) to initialize the candidate solutions, as PrimRST tends to generate low-diameter trees which are more likely to satisfy the max-hop constraint. The PrimRST algorithm uses *Prim's* scheme to greedily create a spanning tree from a start node by adding an adjacent node at random, regardless of its weight. Moreover, we adapt PrimRST to consider max-hop constraint by maintaining nodes' depth while creating a tree. We call this algorithm as *HCPrimRST* (Algorithm 2). Applying HCPrimRST with max-hop constraint may lead to an invalid, non-connected structure. The initialization is thus divided into two phases. The first phase initializes edges $T = \{(0, v) \in E | v \in R\}$ and runs the algorithm with max-hop constraint; and in the second phase, we relax its constraint and continue to build the tree obtained from the first phase to get a valid connected tree.

Algorithm 2 HCPrimRST

Input: The set of initialized edges T, set of vertices V, set of potential edges E, max-hop constraint $\hbar$.
Output: The set of used edges T.

$C \leftarrow \{u, v | (u, v) \in T\}$ # set of connected nodes
$d \leftarrow$ depth of vertices in partial tree T # by dfs from root 0
$A \leftarrow \{(u, v) \in E | u \in C, v \notin C\}$ # eligible edges
while $A \neq \emptyset$ **do**
  Choose $(u, v) \in A$ at random
  $A \leftarrow A \setminus (u, v)$
  if $v \notin C$ **and** $d_u < \hbar$ **then**
    $T \leftarrow T \cup (u, v)$
    $C \leftarrow C \cup \{v\}$
    $A \leftarrow A \cup \{(v, w) \in E | w \notin C\}$ # add v's adjacency to A
    $d_v \leftarrow d_u + 1$
  end if
end while

4.3.3 Crossover operator

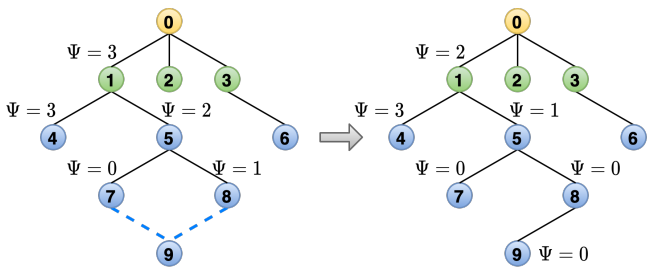


Figure 4.3.1: An example of calculating maximum number of children. Dashed lines denotes potential edges

For crossover, a naive approach is to apply the HCPrimRST algorithm directly to create an offspring T_{cr} from a combined graph $G_{cr} = (V, E_{T_1} \cup E_{T_2})$, where T_1, T_2 are the parental trees. However, due to its random nature, this algorithm might create many infeasible or ineffective offsprings. We propose an energy-aware modification of the HCPrimRST-based crossover to prioritize the offspring's structures that use less energy power than their parents while maintaining the diversity of the offspring.

Following the energy model (4.3), we notice that the dissipated power of a node is affected by three factors, including the number of packets it carries (η_u) (descendant nodes), whether it is a sensor or relay ($\zeta_u \in \{0, 1\}$), and the distance of the transmit-receive pair to its parent ($\xi(d)$):

$$\tilde{E}_u = \eta_u(\tilde{E}_r + \tilde{E}_t(d)) + \zeta_u \tilde{E}_t(d) \tag{4.6}$$

$$\Leftrightarrow \quad \eta_u = \frac{\tilde{E}_u - \zeta_u \tilde{E}_t(d)}{\tilde{E}_r + \tilde{E}_t(d)}. \tag{4.7}$$

Equation (4.7) suggests that if we know the maximum energy a node can use and its parent, we also know the maximum number of packets it can carry. Let us assume that the network has an energy limit of $\tilde{E}_{max}$. Before adding an edge (u, v) where $u \in C, v \notin C, d_u < \hbar$, we can check if this causes the network to exceed the energy limit by tracking the number of descendants that a node in the partial tree can receive.

We denote Ψ_u as the incumbent number of descendants a node can carry without exceeding $\tilde{E}_{max}$ and P_u as a set of u's parent nodes and itself. An edge (u, v) is considered valid if $\Psi_u > 0$. Connecting (u, v) reduces the capacity of the nodes in P_u:

$$\Psi_t = \Psi_t - 1 \quad \forall t \in \mathrm{P}_u. \tag{4.8}$$

Thus, Ψ_v is updated using the following function:

$$\Psi_v = \min\{\eta_v, \Psi_u\}, \tag{4.9}$$

where η_v is calculated as Equation (4.7) with the energy limit $\tilde{E}_{max}$. For example, with the graph presented in Figure 4.3.1, two potential edges connect node 9 to the partial tree: $(7, 9)$ and $(8, 9)$. However, as $\Psi_7 = 0$, connecting the edge $(7, 9)$ inevitably causes one of the parents of node 7 to be exhausted. Thus, the edge $(8, 9)$ is prioritized in this case. Note that adding a child affects all parent nodes. Thus, the maximum number of descendants of a node does not exceed its parents; that is

$$\Psi_u = \min_{t \in \mathrm{P}_u} \Psi_t. \tag{4.10}$$

To reduce computation, an update of node u (Equation 4.10) is only executed when it is necessary. In the example in Figure 4.3.1, Ψ_4 should be updated to 2. However, it is unnecessary until another edge of node 4 is involved. The operator can be done in $O(n\hbar)$.

4.3.4 Mutation operators

One of the most widely used mutations for tree-based problems is *edge insertion mutation* (Raidl, 2000). The tree is mutated by randomly adding a non-tree edge (creating a cycle) and then randomly deleting a tree edge from the created cycle (removing the cycle). This mutation can be applied to most tree-based problems. However, exploring neighbors

randomly may produce many invalid or lower-quality solutions, especially when approaching local optimal solutions.

We propose two problem-specific mutations that improve the algorithm's convergence speed. The first mutation targets energy usage, while the second aims to reduce the number of used relays. These mutations are used simultaneously and chosen randomly for each offspring with predefined probabilities.

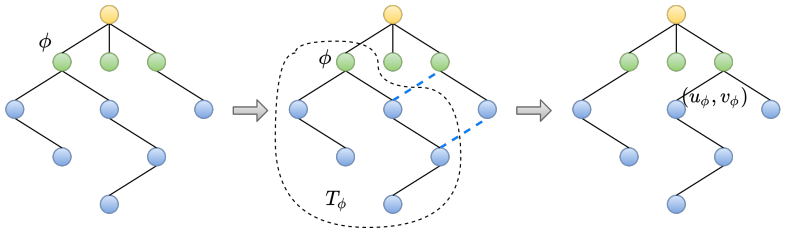


Figure 4.3.2: An example of the energy-oriented mutation. The dashed blue lines denote the potential added edges.

Energy-oriented mutation Let us look closer into the edge insertion mutation. A mutated solution can be represented by its direct parent and a pair of added edge $e^+ = (u^+, v^+)$ and deleted edge $e^- = (u^-, v^-)$. Both edges are chosen at random, most of which lead to a worse solution or violate the max-hop constraint. However, we can leverage each node's maximum number of descendants Ψ (Section 4.3.3) to find and prioritize pairs of edges that ensure a better solution.

Let us denote ϕ as the node using the most energy. Recall that the energy usage of ϕ is composed of two factors: the number of descendants of ϕ and the distance between ϕ and its parent. Reducing the maximum energy usage of the network requires one of two factors to decrease. Consider the subtree $T_\phi = (V_\phi, E_\phi)$, which includes ϕ and its children. We define potential added edge (u, v) as the edge that connects a part of subtree T_ϕ to the remaining subtree $T \setminus T_\phi$ and satisfies the following constraints:

$$u \in V_\phi, \quad v \in V \setminus V_\phi, \quad \eta_u \leq \Psi_v, \quad h_u + d_v + 1 \leq \hbar,$$

where d_u, h_u, η_u is depth (from root to u), maximum hop (from u to farthest leaf) and the number of children of node u, respectively. Also, we refer to p_u as the parent of u in the parental tree. Then, replacing the edge (u, p_u) by (u, v) ensures that the newly constructed tree satisfies the max-hop constraint and has lower energy usage in node ϕ. Figure 4.3.2 shows an example of this process.

When combining the above heuristic with edge insertion mutation, we find a set of potential edge pairs in the tree. If this set is not empty, we randomly choose one pair and create a mutated tree. Otherwise, the original edge insertion mutation is used to complete the mutation operator. Algorithm 3 presents this energy-oriented mutation procedure. The complexity of this mutation is $O(max(n\hbar, m))$.

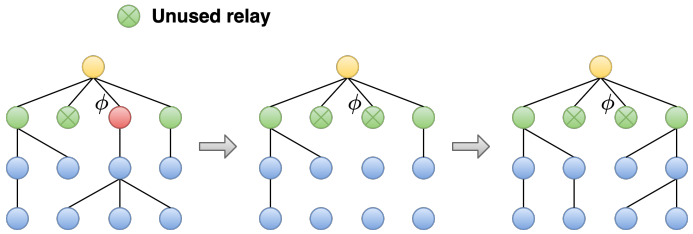


Figure 4.3.3: An example of the relay-oriented mutation

Algorithm 3 The energy-oriented mutation algorithm

Input: Edge-set of parental tree T, set of vertices V, set of potential edges E, energy limit $\tilde{E}_{max}$, max-hop constraint $\hbar$.

Output: The set of edges in the offspring T.

Find a node that uses most energy ϕ in parental tree T

Run a depth-first search in parental tree T to calculate the set of potential edges $F = \{(u,v) \in E | u \in V_\phi \wedge v \in (V \setminus V_\phi) \wedge \eta_u \leq \Psi_v \wedge h_u + d_v + 1 \leq \hbar\}$

if $F \neq \emptyset$ **then**

Choose an edge $(u_\phi, v_\phi) \in F$ at random

$T \leftarrow T \setminus (u_\phi, p_{u_\phi})$ # p_{u_ϕ} is parent of u_ϕ in parental tree

$T \leftarrow T \cup (u_\phi, v_\phi)$

else

Choose an edge $(u^+, v^+) \in E \setminus T$ at random

$T \leftarrow T \cup (u^+, v^+)$

Find a set of edges C in the created cycle

Choose an edge $(u^-, v^-) \in C$

$T \leftarrow T \setminus (u^-, v^-)$

end if

Relay-oriented mutation Due to the tendency toward star-like structures, PrimRST and HCPrimRST uses more relays on average than KruskalRST and RandWalkRST (Figure 4.3.4). In comparison, both crossover and energy-oriented mutation focus on optimizing the energy objective. The optimization of the number used relays depends on the distribution of the random-tree algorithm implemented in the initialization and crossover operator. We propose a more direct mutation strategy for reducing the number of used relays: one random relay from the parental tree is disabled, while its children (sensors) are connected to the remaining relays. The HCPrimRST algorithm is first applied with an energy limit as in the crossover operator. Constraints are later relaxed to ensure that all sensors are connected. An example of this mutation is shown in Figure 4.3.3, and a pseudocode is provided in Algorithm 4. This mutation can be done in $O(n\hbar)$.

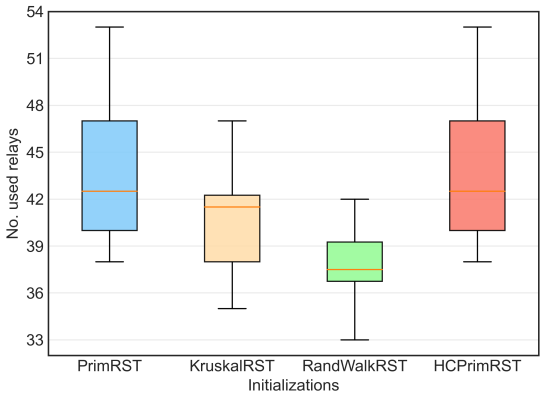


Figure 4.3.4: The distribution the number of used relays generated by different random-tree algorithms on a graph with 100 relays and 100 sensors.

Algorithm 4 The relay-oriented mutation algorithm

Input: Edge-set of parental tree T, set of vertices V, set of potential edges E, energy limit $\tilde{E}_{max}$, max-hop constraint $\hbar$.

Output: The set of edges in the offspring T.

1: Find a set of used relays Φ in the parental tree T
2: Choose one relay $\phi \in \Phi$ at random.
3: $\Phi \leftarrow \Phi \setminus \{\phi\}$
4: Find a set of children node V_ϕ of ϕ
5: $T \leftarrow T \setminus \{(u,v) \in T | u \in V_\phi \vee v \in V_\phi\}$
6: $E \leftarrow E \setminus \{(u,v) \in E | u \in (V_r \setminus \Phi) \vee v \in (V_r \setminus \Phi)\}$ # disable unused relays
7: HCPrimRST($T, S \cup \Phi, E, \hbar, \tilde{E}_{max}$)
8: HCPrimRST($T, S \cup \Phi, E, \hbar, \infty$)
9: HCPrimRST($T, S \cup \Phi, E, \infty, \infty$)

4.4 Experiments

4.4.1 Experiment settings

In our experimental studies, network parameters are set as in Tam et al. (2020) where network constants are as in Table 4.2.1 with parameters $l = 4000$ and $d_0 = \epsilon_{md} \div \epsilon_{fs}$. We also assume that all sensors and relays have the same radius $r = 25$ meters. All algorithms were implemented in Python using the GeneticPython (Bui, 2020) and NumPy (Van Der Walt et al., 2011) frameworks[1]. All experiments are performed in a single Intel Xeon(R) E-2124G 3.40 GHz CPU with 16 GB RAM running on Ubuntu Linux 16.04. No extra parallelization apart from the default NumPy acceleration is used.

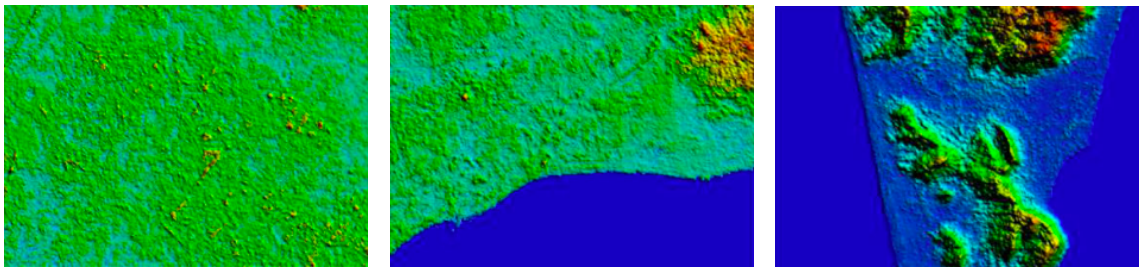
4.4.2 Datasets

We generate 18 network instances according to previous works (Hai et al., 2017, Tam et al., 2018, 2020). Three real 3D terrain datasets in Vietnam (Figure 4.4.1) are used as the area to place sensors and relays. All terrains are defined according to the Digital

[1]Source codes and datasets: `https://github.com/ngocbh/nebp_wsn`

Table 4.4.1: Description of network instances. The last column refers to the density of the communication graph.

	Instance	Node distribution	Terrain	Terrain size	No. relays	No. sensors	Density
$S1$	NIn1	Gaussian	T1	200×200	20	20	0.53
	NIn2	Gaussian	T2	200×200	20	20	0.61
	NIn3	Gaussian	T3	200×200	20	20	0.53
	NIn4	Uniform	T1	200×200	20	20	0.53
	NIn5	Uniform	T2	200×200	20	20	0.53
	NIn6	Uniform	T3	200×200	20	20	0.48
$S2$	NIn7	Gaussian	T1	200×200	40	40	0.28
	NIn8	Uniform	T2	200×200	40	40	0.1
	NIn9	Gaussian	T2	500×500	100	100	0.19
	NIn10	Uniform	T3	500×500	100	100	0.17
	NIn11	Gaussian	T3	1000×1000	200	200	0.52
	NIn12	Uniform	T1	1000×1000	200	200	0.15
$S3$	NIn13	Gaussian	T1	200×200	40	80	0.2
	NIn14	Uniform	T2	200×200	40	80	0.18
	NIn15	Gaussian	T2	500×500	100	200	0.1
	NIn16	Uniform	T3	500×500	100	200	0.09
	NIn17	Gaussian	T3	1000×1000	200	400	0.28
	NIn18	Uniform	T1	1000×1000	200	400	0.28



(a) T1: Ho Chi Minh (b) T2: Vung Tau (c) T3: Phu Quoc

Figure 4.4.1: Height heatmaps of terrains.

Elevation Model (DEM) standard. The sensors and potential relay nodes are deployed in each terrain according to two distributions (Gaussian and Uniform).

We split the data into three sets: $S1$, $S2$, and $S3$. $S1$ includes 6 instances with 20 RNs and 20 SNs. $S2$ contains 6 instances with increasing SNs in the network. $S3$ uses the same settings as $S2$ with twice the SNs in the network. The usage of these sets is discussed in detail in Subsection 4.4.5. Details of the network instances are shown in Table 4.4.1.

4.4.3 Performance metrics

Comparing Pareto fronts is not very straightforward. No single metric can best cover all aspects (cardinality, convergence, diversity) (Audet et al., 2020, Konstantinidis and Yang, 2011, Riquelme et al., 2015). Thus, we report the following five metrics.

- **Inverted Generational Distance (IGD)** (Coello and Cortés, 2005): Given an op-

timal Pareto front (PF) P, IGD of a approximation set S is calculated as:

$$IGD(S,P) = \frac{1}{|P|}\left(\sum_{p\in P} d_p^l\right)^{\frac{1}{l}},$$

where $d_p = \min_{s\in S} \|f(s) - f(p)\|$ and $l = 2$ (in general). IGD can capture both the convergence and diversity of approximation PFs. However, this metric cannot be used without an optimal PF.

- **Hypervolume (HV)** (Zitzler and Thiele, 1999): As described in Audet et al. (2020), the hypervolume indicator is the volume of the space dominated by the Pareto front approximation S and delimited from above by a reference point $r \in R^m$ such that for all $s \in S, s \prec r$. The hypervolume is given by:

$$HV(S,r) = \lambda_m\left(\bigcup_{s\in S}[s,r]\right),$$

where λ_m is the m-dimensional Lebesgue measure. In a bi-objective problem ($m = 2$), hypervolume can easily be obtained in linear time.

- **Convergence of two sets (C)** (Zitzler and Thiele, 1998): This metric is widely used to capture the convergence of two approximation sets. It is defined by the ratio of a set dominated by others divided by its cardinality:

$$C(A,B) = \frac{|\{b \in B | \exists a \in A : b \prec a\}|}{|B|}.$$

If $C(A,B) = 1$, all solutions of B are dominated by solutions of A. Note that we have to compute both $C(A,B)$ and $C(B,A)$ since their sum is not always equal to 1.

- **Delta-metric (Δ)** (Deb et al., 2002): This is a unary metric used to measure a PF's diversity. In bi-objective problem, $\Delta(S)$ of a PF S is defined as:

$$\Delta(S) = \frac{d_f + d_l + \sum_{i=1}^{|S|-1}|d_i - \bar{d}|}{d_f + d_l + |S|\bar{d}},$$

where d_f and d_l are the Euclidean distances between the extreme solutions in one objective and the boundary solutions of S. d_i is the Euclidean distance between consecutive solutions of approximation set S and $\bar{d}$ is the mean of d_i. The smaller value of $\Delta(S)$ gives a better spread of the PF.

- **Cardinality ($ONVG$)** (Schott, 1995): This is a straightforward metric computing a PF's cardinality

$$ONVG(S) = |S|.$$

Since the NEBP is a bi-objective problem, one of which is discrete with a low range, $ONVG$ is essential to measure the spread of PFs.

4.4.4 Baselines

We compare our proposed method with five algorithms, including different encoding methods and search operators that are widely used in tree encoding problems:

- **HMOEA**: We adopt a hybrid multi-objective evolutionary algorithm (HMOEA) proposed in Prakash et al. (2020) for bi-Objective Minimum Diameter-Cost Spanning Tree (bi-MDCST) problem to tackle the NEBP. A permutation encoding with a heuristic CBTC-based decoder is proposed to represent several solutions of varying relay's constraint in which the *order crossover* and *swap mutation* are used for reproduction.
- **Prufer encoding (Prufer)**: Prufer encoding is used in conjunction with two common search operators in integer chromosomes: *uniform crossover* and *swap mutation*. To reduce the infeasible ratio caused by incomplete graphs, we repair the decoded edges using only valid edges with KruskalRST to complete the candidate tree.
- **NetKeys encoding (Netkeys)**: In this scheme, we encode a tree using general Network Random Keys with *uniform crossover* and *swap mutation*, as suggested in Rothlauf et al. (2002).
- **Edge-set and PrimRST (Prim)**: This algorithm is based on the application of edge sets to the degree-constraint minimum spanning tree problem (d-MST) as presented in Raidl and Julstrom (2003). In the recombination process, we use *PrimRST* to create offspring from the parents' edges. In the mutation, we use the *edge insertion mutation*, as described in Section 4.3.4.
- **Edge-set and KruskalRST (Kruskal)**: All settings are the same as in the Prim-based approach, but *KruskalRST* is used instead of *PrimRST* in the recombination.
- **Edge-set and Guided Prim (GPrim):** This is our proposed algorithm.

In these approaches, the first three algorithms (HMOEA, Prufer, and NetKeys) are indirect representations, while the rest (Prim, Kruskal, and GPrim) are direct representations. Leveraging the discreteness of one of the objectives, HMOEA maintains an external population to store the Pareto front. Each offspring will be evaluated in different constraints of relays to update the Pareto front as well as the main population, where the tournament of size three is used to select the parents. Meanwhile, the remaining algorithms are applied to the NSGA-II structure with *binary tournament selection*. Note that our methods can also apply to other multi-objective optimizations (MOOs) algorithm structures such as MOEA/D (Cheng et al., 2015), SPEA2 (Zitzler et al., 2001). However, such combinations are reserved for future works.

4.4.5 Results and discussions

We design four main experiments to study the behavior of the approaches through four aspects:

- **Efficiency**: In this experiment, we use six instances in $S1$ and find an approximation of optimal PF for each instance by combining the results of the six algorithms with a vast number of generations. We later compare the algorithms on the IGD metric, as well as their convergence process through each generation.
- **Scalability**: This experiment uses 12 instances in sets $S2$ and $S3$, to study the behavior of algorithms as network complexity increases.
- **Sparsity of feasible solution**: This experiment aims to investigate the ability to handle sparse solution spaces by reducing the max-hop constraint on each instance in all datasets.

- **Density of the network**: Various communication ranges are used to examine the impact of the graph's density on the algorithms.

Table 4.4.2: The max-hop constraint ($\hbar$) on each type of dataset. All instances are ensured to have valid solutions.

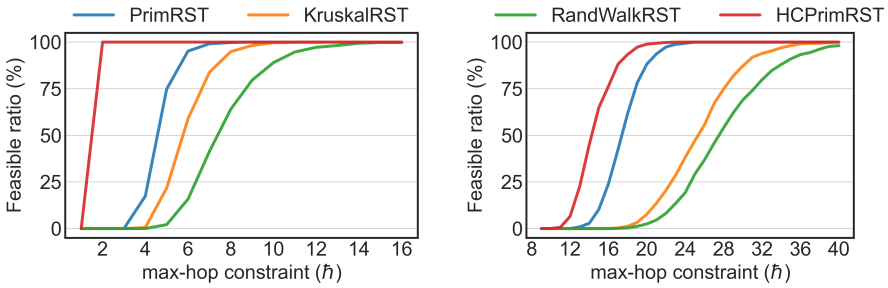
	Type 1	*Type 2*	*Type 3*	*Type 4*
$\hbar$	6	8	12	16

Table 4.4.3: The operator's probability of each algorithm. pc is the crossover probability and pm is the mutation probability. For GPrim, pm represents a pair of energy-oriented and relay-oriented mutation probability

	HMOEA	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
pc	0.6	0.9	0.9	0.9	0.9	0.5
pm	0.4	0.5	0.1	0.5	0.5	(0.5, 0.5)

Parameter selection. We set the population size to 100 while the number of generations is set to 100. The max-hop constraint is set depending on the complexity of the network instance (see Table 4.4.2). Each experiment is performed over 10 independent runs with different seeds.

Since each algorithm has different crossover and mutation operators, we perform a grid search to select crossover and mutation probability. Five random instances are generated, on which the probabilities are searched independently for each algorithm. Table 4.4.3 shows the resulting settings.



(a) *NIn9* - 100 SNs and 100 SNs (b) *NIn15* - 100 SNs and 200 SNs

Figure 4.4.2: Feasible ratio of the population on various max-hop constraints.

Initialization analysis. We compare the ratio of feasible solutions constructed by the proposed initialization HCPrimRST with other standard initializations PrimRST, KruskalRST, and RandWalkRST. The result is shown in Figure 4.4.2. We run four initializations (PrimRST, KruskalRST, RandWalkRST, HCPrimRST) on two instances (*NIn9* and *NIn15*) with various max-hop constraint $\hbar$. Each algorithm is invoked 1000 times for each value of $\hbar$. Figure 4.4.2 shows the feasible solution ratio of the initializations on a different value of $\hbar$. The HCPrimRST gives the best feasible ratio in all max-hop constraints. The feasible ratio of the remaining methods (PrimRST, KruskalRST, and RandWalkRST) is inversely

Table 4.4.4: Performance of competing algorithms on the set $S1$. Bold values indicate the best values.

(a) Inverted Generational Distance (IGD)

Instance	*HMOEA*	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn1	0.039 ± 0.007	0.037 ± 0.005	0.039 ± 0.006	0.033 ± 0.008	0.036 ± 0.004	**0.000** ± 0.000
NIn2	0.043 ± 0.011	0.038 ± 0.004	0.038 ± 0.005	0.033 ± 0.007	0.036 ± 0.007	**0.000** ± 0.000
NIn3	0.035 ± 0.006	0.036 ± 0.005	0.033 ± 0.007	0.034 ± 0.006	0.035 ± 0.003	**0.000** ± 0.000
NIn4	0.033 ± 0.007	0.036 ± 0.006	0.036 ± 0.007	0.034 ± 0.007	0.039 ± 0.005	**0.000** ± 0.000
NIn5	0.034 ± 0.004	0.038 ± 0.007	0.039 ± 0.004	0.032 ± 0.005	0.037 ± 0.007	**0.000** ± 0.000
NIn6	0.035 ± 0.005	0.036 ± 0.004	0.037 ± 0.006	0.033 ± 0.007	0.034 ± 0.006	**0.000** ± 0.000

(b) Cardinality ($ONVG$)

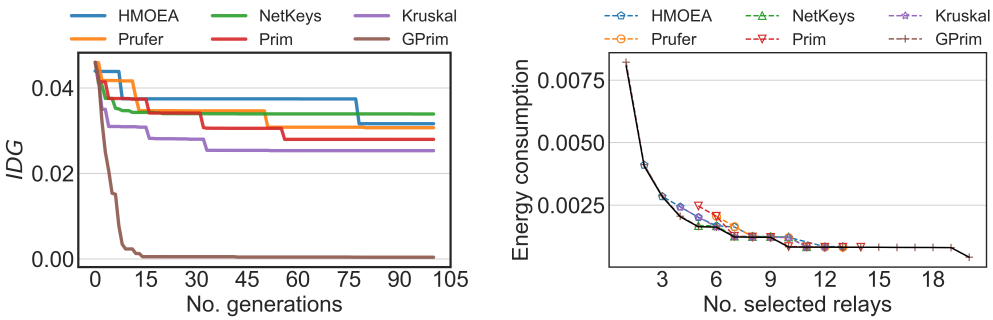
Instance	*HMOEA*	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn1	9.90 ± 0.831	7.90 ± 1.375	8.30 ± 1.269	9.90 ± 2.256	8.90 ± 0.943	**20.00** ± 0.000
NIn2	9.40 ± 1.281	8.00 ± 1.095	8.60 ± 1.281	9.60 ± 1.744	9.40 ± 1.625	**20.00** ± 0.000
NIn3	10.30 ± 0.900	8.00 ± 1.183	10.40 ± 1.562	9.30 ± 1.616	9.30 ± 0.900	**20.00** ± 0.000
NIn4	10.30 ± 1.100	8.40 ± 1.428	8.70 ± 1.269	9.10 ± 1.700	8.30 ± 1.100	**20.00** ± 0.000
NIn5	10.30 ± 1.005	8.20 ± 1.400	8.70 ± 0.781	9.70 ± 1.552	8.60 ± 1.625	**20.00** ± 0.000
NIn6	10.70 ± 0.781	7.50 ± 1.025	8.70 ± 1.487	10.00 ± 2.449	9.70 ± 1.900	**20.00** ± 0.000

proportional to the average diameter in the results shown in Raidl and Julstrom (2003). For example, when $\hbar = 14$, HCPrimRST produces valid solutions 45% of the time. This metric for PrimRST is 2.8%. KruskalRST and RandWalkRST produced no feasible solutions on average. If KruskalRST or RandWalkRST was applied in this case, it might leave the algorithm with no valid solutions in the first few generations.

Based on the above results, in the following experiments, HCPrimRST will be used as the initialization for all compared algorithms.

Efficiency evaluation. In this experiment, we look into each algorithm's efficiency compared to an approximation of optimal PF. We use six small instances in the set $S1$. Optimal PF is approximated by running each algorithm 100 times with different seeds, each with 1000 generations. The best solutions are combined to generate a final approximation PF.

Table 4.4.4 summarizes the results of competing algorithms through three metrics (IGD, and $ONVG$). GPrim outperforms HMOEA, Prufer, NeyKeys, Prim, and Kruskal with regard to both convergence and diversity. GPrim's PF approaches the approximation PF in most cases. Between standard direct and indirect representations, Prufer showed the worst results in the aforementioned aspects, while NetKeys and HMOEA provide better results than standard direct representations (Prim, Kruskal). In Figure 4.4.3, we show the convergence profiles of IGD values on network instance *NIn1*. GPrim again shows faster convergence than the remaining approaches, while the standard direct representations (Prim and Kruskal) show promising results.



(a) IGD metric over generations

(b) The Pareto-fronts comparison where the black line is the approximate optimal PF

Figure 4.4.3: Comparison of five algorithms on *NIn1*.

Considering the performance in the number of non-dominated solutions, GPrim consistently finds the maximum value of $OVNG$. This result demonstrates the effectiveness of the relay-oriented mutation. HMOEA, NetKeys, Prim, and Kruskal perform similarly and fluctuate around 9 solutions for each instance. Meanwhile, Prufer performs worst on $OVNG$.

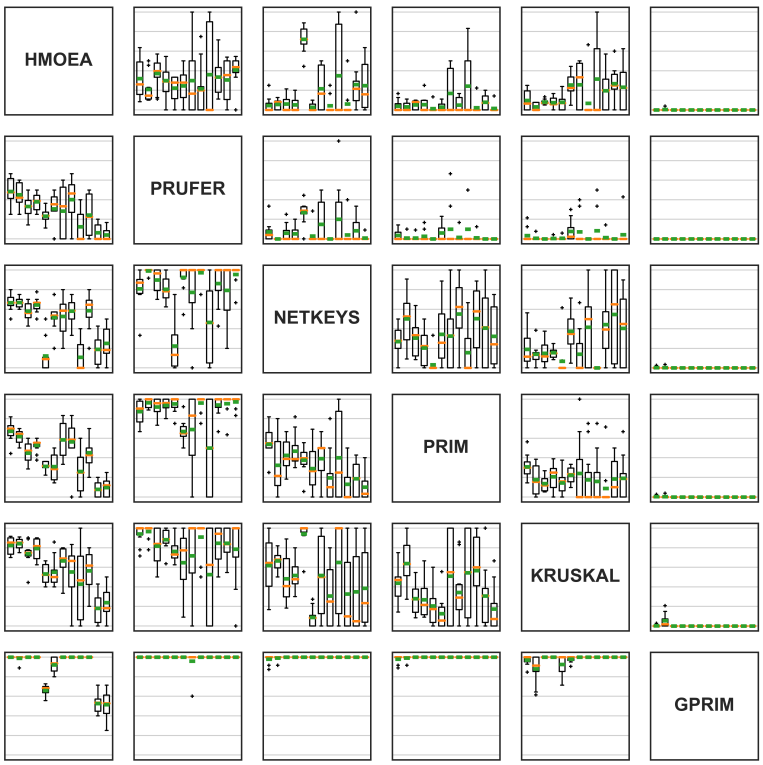


Figure 4.4.4: Box plots for C-metric. The rectangle at row A and column B represent $C(A, B)$. Each rectangle includes 12 box plots (left to right) corresponding to 12 instances (*NIn7* to *NIn18*). C-metric values are scaled to [0,1].

Table 4.4.5: Performance of competing algorithms on testsets $S2$ and $S3$. Each table shows the results on one metric and bold values indicate the best value.

(a) Hypervolume (HV)

Instance	*HMOEA*	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn7	0.879 ± 0.001	0.763 ± 0.013	0.774 ± 0.029	0.765 ± 0.024	0.858 ± 0.015	**0.930** ± 0.001
NIn8	0.854 ± 0.002	0.716 ± 0.021	0.771 ± 0.015	0.740 ± 0.026	0.836 ± 0.024	**0.907** ± 0.005
NIn9	0.909 ± 0.001	0.737 ± 0.015	0.788 ± 0.020	0.733 ± 0.013	0.890 ± 0.019	**0.964** ± 0.006
NIn10	0.910 ± 0.001	0.750 ± 0.011	0.806 ± 0.024	0.753 ± 0.006	0.897 ± 0.022	**0.967** ± 0.000
NIn11	**0.930** ± 0.000	0.602 ± 0.015	0.709 ± 0.025	0.609 ± 0.007	0.764 ± 0.028	0.780 ± 0.017
NIn12	0.922 ± 0.001	0.762 ± 0.011	0.797 ± 0.023	0.738 ± 0.008	0.905 ± 0.016	**0.941** ± 0.021
NIn13	0.788 ± 0.007	0.688 ± 0.018	0.735 ± 0.021	0.740 ± 0.014	0.765 ± 0.019	**0.843** ± 0.001
NIn14	0.751 ± 0.019	0.649 ± 0.022	0.733 ± 0.020	0.725 ± 0.025	0.741 ± 0.015	**0.885** ± 0.006
NIn15	0.741 ± 0.025	0.608 ± 0.038	0.621 ± 0.030	0.671 ± 0.016	0.719 ± 0.046	**0.899** ± 0.001
NIn16	0.896 ± 0.002	0.744 ± 0.010	0.810 ± 0.009	0.762 ± 0.018	0.825 ± 0.016	**0.959** ± 0.001
NIn17	**0.915** ± 0.002	0.575 ± 0.010	0.708 ± 0.032	0.604 ± 0.008	0.700 ± 0.024	0.814 ± 0.015
NIn18	**0.914** ± 0.002	0.556 ± 0.009	0.706 ± 0.018	0.577 ± 0.017	0.697 ± 0.026	0.805 ± 0.011

(b) Delta-metric (Δ)

Instance	*HMOEA*	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn7	0.775 ± 0.028	0.878 ± 0.019	0.892 ± 0.023	0.868 ± 0.020	0.829 ± 0.012	**0.599** ± 0.002
NIn8	0.776 ± 0.031	0.870 ± 0.030	0.873 ± 0.020	0.870 ± 0.019	0.842 ± 0.037	**0.699** ± 0.048
NIn9	0.826 ± 0.017	0.937 ± 0.017	0.941 ± 0.015	0.945 ± 0.017	0.923 ± 0.025	**0.758** ± 0.032
NIn10	0.825 ± 0.022	0.940 ± 0.016	0.944 ± 0.015	0.945 ± 0.010	0.917 ± 0.028	**0.779** ± 0.015
NIn11	0.799 ± 0.017	0.962 ± 0.012	0.948 ± 0.026	0.975 ± 0.016	0.955 ± 0.020	**0.762** ± 0.010
NIn12	0.885 ± 0.028	0.956 ± 0.009	0.974 ± 0.013	0.971 ± 0.009	0.936 ± 0.013	**0.785** ± 0.012
NIn13	**0.882** ± 0.028	0.940 ± 0.040	0.947 ± 0.025	0.953 ± 0.031	0.950 ± 0.033	0.935 ± 0.004
NIn14	**0.874** ± 0.048	0.936 ± 0.034	0.947 ± 0.020	0.921 ± 0.036	0.940 ± 0.035	0.947 ± 0.027
NIn15	**0.896** ± 0.036	0.960 ± 0.022	0.975 ± 0.033	0.969 ± 0.025	0.979 ± 0.023	0.960 ± 0.006
NIn16	**0.851** ± 0.033	0.945 ± 0.019	0.985 ± 0.012	0.967 ± 0.013	0.970 ± 0.012	0.966 ± 0.012
NIn17	**0.878** ± 0.019	0.964 ± 0.009	0.991 ± 0.022	0.983 ± 0.008	0.968 ± 0.016	0.946 ± 0.008
NIn18	**0.877** ± 0.047	0.963 ± 0.012	0.982 ± 0.016	0.987 ± 0.010	0.993 ± 0.011	0.945 ± 0.007

(c) Cardinality ($ONVG$)

Instance	*HMOEA*	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn7	10.30 ± 1.100	10.00 ± 1.483	9.60 ± 1.428	11.00 ± 2.236	14.00 ± 1.095	**38.00** ± 0.000
NIn8	9.50 ± 1.285	9.80 ± 1.887	10.80 ± 1.327	10.00 ± 1.183	12.90 ± 2.948	**27.70** ± 0.900
NIn9	8.50 ± 1.025	10.90 ± 2.343	11.80 ± 2.040	10.90 ± 1.221	15.20 ± 2.561	**71.80** ± 3.572
NIn10	8.80 ± 0.872	10.90 ± 1.972	11.10 ± 2.948	10.30 ± 1.487	17.70 ± 2.934	**69.60** ± 3.323
NIn11	10.40 ± 0.917	9.70 ± 1.900	17.50 ± 3.170	9.80 ± 3.400	16.30 ± 2.452	**98.60** ± 1.114
NIn12	6.80 ± 0.872	11.70 ± 1.187	11.00 ± 1.789	10.30 ± 2.193	19.20 ± 2.441	**94.60** ± 3.980
NIn13	**5.00** ± 1.000	3.10 ± 1.640	2.50 ± 0.671	2.40 ± 1.020	2.40 ± 1.020	4.00 ± 0.000
NIn14	5.10 ± 1.446	4.30 ± 1.487	4.30 ± 1.100	6.20 ± 1.778	3.20 ± 0.748	**9.10** ± 0.539
NIn15	3.80 ± 1.536	3.70 ± 1.005	1.90 ± 0.539	3.00 ± 1.414	2.00 ± 0.894	**6.00** ± 0.000
NIn16	7.00 ± 1.732	8.00 ± 1.183	5.70 ± 1.952	6.50 ± 1.910	6.00 ± 1.732	**24.30** ± 1.900
NIn17	6.20 ± 0.872	9.80 ± 1.990	10.10 ± 3.448	5.00 ± 1.897	6.80 ± 2.600	**23.20** ± 2.891
NIn18	6.50 ± 1.500	8.00 ± 2.191	10.40 ± 2.905	4.00 ± 1.844	6.20 ± 2.522	**22.30** ± 3.257

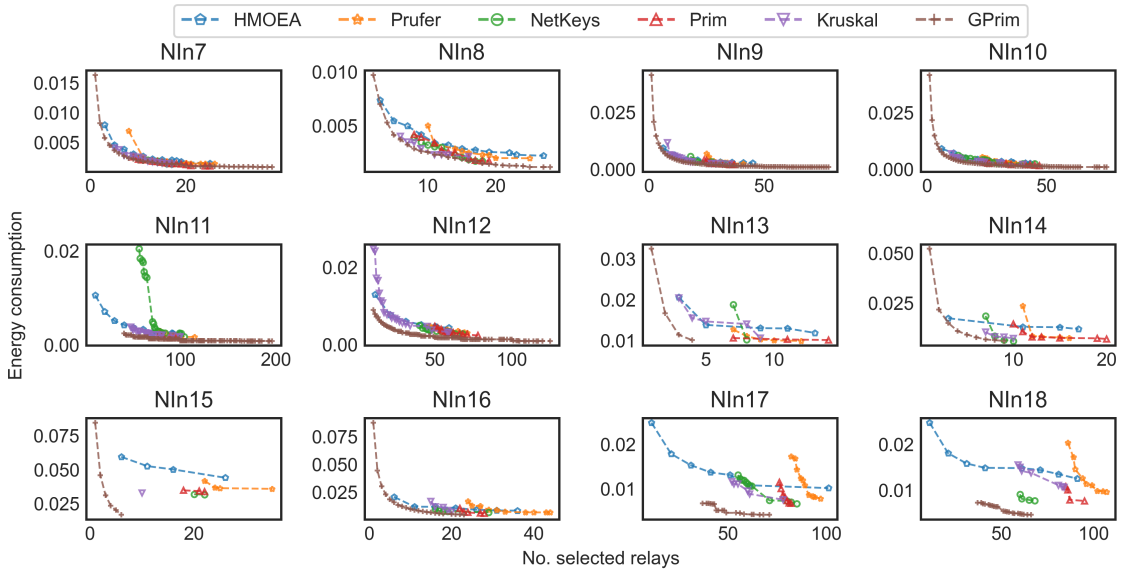


Figure 4.4.5: The comparison of Pareto-front on test set $S2$ (*NIn7* to *NIn12*) and $S3$ (*NIn13* to *NIn18*).

Scalability evaluation. This experiment aims to investigate the behavior of the algorithms on different structures of the graph, especially when the graph size is expanded. We use 12 network instances in sets $S2$ and $S3$ with different distributions and sizes to demonstrate the effectiveness of the algorithm in various topologies of the graph. Since finding the approximation PF costs a massive amount of the computation power on complex network instances, we consider the convergence of two sets (C-metric) and delta metric (Δ) instead of IGD.

In terms of convergence, we use *box plots* similar to Zitzler et al. Zitzler and Thiele (1999) to summarize C-metric values of different runs (see Figure 4.4.4). Results show that GPrim produces dominating PFs over other algorithms in most cases, with C values asymptotic to 1 in most test cases. Specifically, GPrim covers more than 88% on average of the fronts computed by HMOEA, while only 0.03% of which dominates the solutions of GPrim. In several instances (*NIn11, NIn12, NIn17, NIn18*), HMOEA produces some better solutions over GPrim with fewer relays due to the mechanism that decodes and evaluates each individual on different relay constraints. It diversifies the evaluated solutions across different numbers of relays. As a trade-off, HMOEA shows poor energy consumption and C-metric results, even in comparison with NetKeys, Prim, and Kruskal. Comparing GPrim and Kruskal, the PFs of GPrim still dominate the PFs of Kruskal in the test set $S3$, where the number of sensors is twice the number of potential relays. In the set $S2$, Kruskal produces some better solutions over GPrim in energy consumption with small networks (*NIn8*) or use fewer relays in more extensive networks (*NIn12*). However, GPrim still covers, on average, more than 97% of the PFs computed by Kruskal in all network instances. Figure 4.4.5 illustrates the PFs of six algorithms with a specified seed.

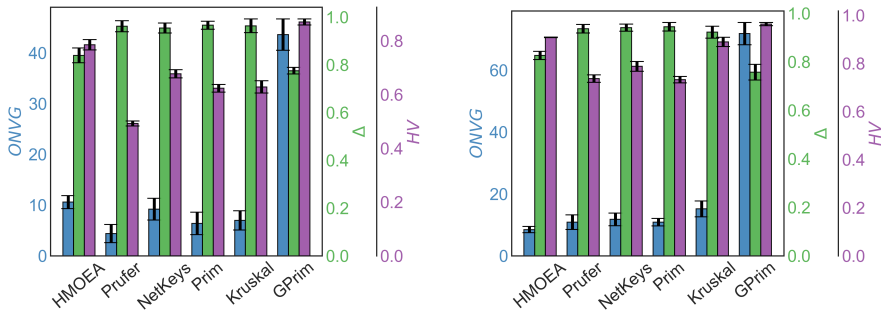
In terms of hypervolume (see Table 4.4.5a), we can observe that GPrim outperforms Prufer, NetKeys, Prim, and Kruskal in all instances, especially on the $S3$ set. Meanwhile, HMOEA produces a better HV metric than GPrim in some instances (*NIn11*, *NIn17*, and *NIn18*) and worse than GPrim in the remaining instances. Note that the superiority of HMOEA in those instances comes from the better spreading of PFs regarding the relay's objective while the convergence of HMOEA is weak as mentioned in C-metric above. Prufer gives the worst result in all test cases among the remaining algorithms. The PFs

obtained by NetKeys are generally better than Prim. Meanwhile, Kruskal outperforms both NetKeys and Prim. However, the results of Kruskal deteriorate in the test set $S3$.

Considering the diversity and spread of PFs, the results on Δ value (see Table 4.4.5b) shows that GPrim and HMOEA produce very diverse non-dominated networks with $\Delta = 0.840$ on average while the average diversity of the HMOEA is $\Delta = 0.845$ in which GPrim performs the best in the second test set $S2$ and HMOEA is the best in the third test set $S3$. The average Delta metric of Kruskal is $\Delta = 0.933$; Prufer is $\Delta = 0.938$; NetKeys and Prim have a very similar result $\Delta = 0.946$. Furthermore, the number of nondominated solutions obtained by GPrim (40 solutions on average) is more than four times better than others. The detail on each network instance is shown in Table 4.4.5c.

In general, the proposed approach outperforms other approaches in all evaluated aspects. HMOEA shows an advantage in the relay's objective resulting in the diversity and spread of PFs. However, as a trade-off, HMOEA provides weak performances in the convergence aspect. The poor result of the Prufer code is predictable due to problems with infeasible structures and low locality, as mentioned in Gottlieb et al. (2001), Rothlauf (2006). The general phenotypic application of PrimRST is generally worse than the NetKeys approach. In comparison, Kruskal can achieve better results than NetKeys in most cases. This result is consistent with the results shown Raidl and Julstrom (2003). However, the results of Kruskal tend to deteriorate in the third test set $S3$.

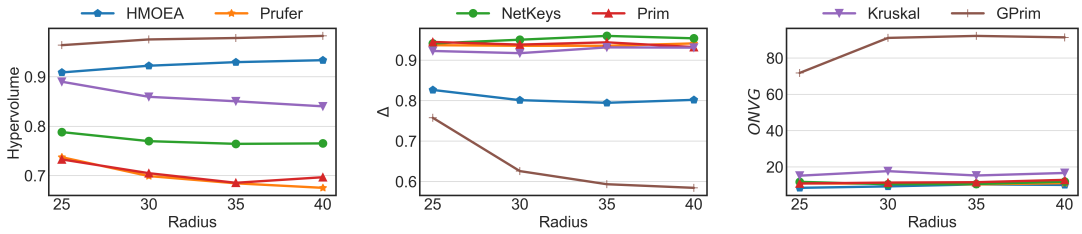
Evaluation on sparse solution spaces. The initialization experiment (Section 4.4.5) showed the relation between the max-hop constraint and the sparsity of solution space and its impact on different random-tree algorithms. Since KruskalRST tends to produce higher-diameter trees than PrimRST, it also produces more infeasible solutions when tightening the max-hop constraint. Consequently, the KruskalRST-based approach also struggles in a sparse solution space.



(a) $\hbar = 2$ (b) $\hbar = 12$

Figure 4.4.6: Performance of competing algorithms in different $\hbar$ on network instance *NIn9*.

For example, we examine five algorithms on the same instance *NIn9* with $\hbar = 2$ and $\hbar = 12$. The result is shown in Figure 4.4.6. With the same settings as in 4.4.5 ($\hbar = 12$), Kruskal gives better results than both NetKeys and Prim. However, when tightening the max-hop constraint ($\hbar = 2$), the result obtained by Kruskal is worse than both NetKeys and Prim. This result is consistent with the deterioration of Kruskal in the test set $S3$ since adding more sensors in the network requires more hops to connect all sensors. Note that the best results are obtained by GPrim in both cases; the gap to other algorithms increases when the feasible solution space becomes smaller.



(a) Hypervolume (b) Delta-metric (c) $ONVG$

Figure 4.4.7: Comparison of hypervolume, delta-metric and $ONVG$ with different communication radius on network instance *NIn9*.

Evaluation on dense network. We vary the communication range from 25 to 40 to examine the impact of the density of the graph on six algorithms. Figure 4.4.7 shows the comparison of hypervolume, Delta-metric, and $ONVG$ with different communication radii on the network instance *NIn9*. The ranking of the algorithms is similar to that of the previous experiments. However, we can observe the different tendencies of algorithms when increasing the radius. Prufer, NetKeys, Prim, and Kruskal deteriorate their results when increasing the communication range, while GPrim and HMOEA tend to produce better results with the dense graph.

Complexity evaluation. We summarize the analysis of the complexity of the five algorithms in Table 4.4.6. The integer search operators on Prufer code only take $O(n)$ in each move. However, the repair process of removing invalid edges takes $O(m)$ in the worst case. Both Prim and Kruskal only take $O(n)$ in crossover and mutation operator and require no repair process. NetKeys encodes all possible edges and requires a sorting algorithm; thus, it costs $O(m \log m)$ in decoding and $O(m)$ in both crossover and mutation. In HMOEA, the decoder uses a CBTC-based heuristic which requires $O(n^2)$ to decode a chromosome for a given relay constraint, while the reproduction costs $O(n)$ for a recombination and $O(1)$ for a mutation. The complexity of GPrim is the same as described in Section 4.3.

Table 4.4.6: Complexity comparison for each algorithm.

	Decoding	Crossover	Mutation	Repair
Prufer	$O(n)$	$O(n)$	$O(n)$	$O(m)$
NetKeys	$O(m \log m)$	$O(m)$	$O(m)$	-
Prim	$O(n)$	$O(n)$	$O(n)$	-
Kruskal	$O(n)$	$O(n)$	$O(n)$	-
HMOEA	$O(n^2)$	$O(n)$	$O(1)$	-
GPrim	$O(n)$	$O(n\hbar)$	$O(max(n\hbar, m))$	-

Table 4.4.7 shows the average running time of five algorithms in three test sets. The running time of GPrim is an acceptable tradeoff considering its various improvements. The fastest algorithm is Prim in most cases. Prufer gives the fastest result in some cases but is often much slower as infeasible structures occur more frequently. Kruskal has the same theoretical complexity as Prim, but requires higher programming constants. Meanwhile, NetKeys and, especially, HMOEA require much computation in large and dense networks.

Table 4.4.7: Average algorithm running time (in seconds).

	HMOEA	*Prufer*	*NetKeys*	*Prim*	*Kruskal*	*GPrim*
NIn1	35.1	19.1	25.0	**17.2**	21.8	23.9
NIn2	39.2	19.3	26.6	**17.2**	21.8	24.0
NIn3	39.6	27.3	34.3	**19.8**	29.8	26.1
NIn4	39.6	28.4	34.5	**19.8**	30.0	26.0
NIn5	39.6	27.9	34.5	**19.8**	30.5	26.2
NIn6	37.1	28.0	34.3	**19.9**	30.5	25.8
NIn7	85.9	**27.9**	39.5	29.7	34.5	42.2
NIn8	36.5	**22.9**	24.0	28.9	32.1	40.6
NIn9	342.2	**63.0**	129.3	68.2	84.4	98.8
NIn10	293.8	**59.0**	114.4	70.8	81.6	98.2
NIn11	3921.3	235.8	1151.3	**195.0**	338.0	249.0
NIn12	1060.3	**131.5**	379.2	158.3	199.4	215.1
NIn13	117.9	**36.0**	55.9	43.1	44.5	76.3
NIn14	101.3	**37.5**	50.5	41.9	46.5	73.6
NIn15	365.8	110.1	154.9	**106.9**	119.8	227.8
NIn16	332.6	125.8	134.4	**106.3**	115.9	178.2
NIn17	4303.1	594.6	1386.9	**287.2**	436.8	459.8
NIn18	4427.1	639.5	1493.7	**297.5**	463.0	478.2

4.5 Discussion

In this chapter, we introduced the Node-Energy Bottleneck problem in multi-hop wireless sensor networks (NEBP) and presented two objectives for the problem: minimizing the number of relay nodes and maximizing the network lifetime. We then developed a phenotype-based multi-objective evolutionary algorithm that simultaneously optimizes both objectives. To enhance the convergence speed of the algorithm, we proposed local heuristics for initialization, crossover, and mutations. We conducted extensive experiments to compare the proposed method with other standard encoding methods. The simulation results demonstrate that the proposed method outperforms the other methods in terms of all measured metrics with a reasonable computation time.

However, one limitation of the proposed scheme is that it assumes that the relay nodes connect directly to the base station. In some applications, such as in large-scale networks or in areas with limited connectivity, this assumption may not hold, which may limit the practicality of the proposed approach. It would be interesting to investigate the effectiveness of the proposed scheme under more realistic network topologies and connectivity scenarios.

Furthermore, in future research, we aim to develop more optimized algorithms for the NEBP problem and extend the model to deal with mobile sensors. Additionally, we plan to investigate the same problem for heterogeneous wireless sensor networks rather than just homogeneous WSNs. Another avenue for future research is to explore hybridizing different encoding methods with different MOO algorithms, which may provide further insights into the impact of representation on each algorithm. The results obtained here highlight the potential of phenotype-based approaches, which may be promising for similar problems.

Chapter 5

A Reinforcement Learning-based Charging Policy in WRSNs

This chapter studies the adaptive charging problem in WRSNs with target coverage and connectivity constraints. The main advantage of the WRSN paradigm compared to other techniques, such as energy-efficient routing and energy harvesting, is the ability to provide continuous and reliable services. The basic idea behind WRSNs is to use a mobile charger (MC) with a high-power battery to move around and wirelessly charge the sensors. However, the main challenge lies in designing a suitable charging strategy for the mobile charger, which should account for uncertainties arising in the network. The existing charging schemes either make a strict assumption about constant energy consumption rates or cannot adapt to unpredictable changes in network topology. To overcome this challenge, we propose a new charging scheme that uses deep reinforcement learning (DRL) to guide the mobile charger adaptively. This approach allows the mobile charger to adjust to spontaneous changes in the network topology. The empirical results show that our method outperforms existing charging schemes by a significant margin.

5.1 Introduction

The real-world deployment of WSNs must fulfill many quality-of-service (QoS) requirements, in which coverage and connectivity are often considered two paramount factors (Tripathi et al., 2018). The coverage specifies how well the sensors monitor the areas or targets of interest, while the connectivity relates to the ability to transmit the sensing data from the sensors to the BS. Ensuring coverage and connectivity is critical since, in many applications, the network is required to monitor and analyze the targets or areas continuously (Zhao and Gurusamy, 2008).

However, maintaining the network for continuous surveillance is an enormous challenge due to the energy restriction of the sensors–that the sensors are often equipped with low-cost, low-power batteries (Akyildiz et al., 2002). When the battery is fully consumed, a sensor can no longer monitor the targets or relay the data; thus, the network may become fragmented, and the data from some parts of the sensing field may no longer be extracted. Furthermore, sensor networks are often implemented on a wide scale in potentially hazardous terrain that is difficult for people to access (e.g., battlefields, underground). In such circumstances, it is difficult to replace the sensors' batteries.

Traditional approaches focus on conserving energy via optimizing sensor functioning,

such as data reduction (Goyal et al., 2019), sleep/wakeup schemes (Haimour and Abu-Sharkh, 2019), and energy-efficient routing (Raj et al., 2019). However, this approach is only able to extend the sensors' lifetime for a certain amount of time. The battery will eventually be exhausted if there is no external source supplying the sensors. An alternative solution is to deploy an energy harvester or scavenger inside each sensor to convert energy from an external source (e.g., solar, thermal, wind) (Adu-Manu et al., 2018). Nevertheless, this technique dramatically depends on an ambient source that is usually unstable and uncontrollable.

Recent progress in wireless power transmission technology based on electromagnetic waves (Kurs et al., 2007, Lu et al., 2015) has given rise to a novel scheme, namely *wireless rechargeable sensor networks* (WRSNs) (He et al., 2012), for energizing the sensors. The idea is to employ a (or multi-) mobile charger (MC) equipped with a high-capacity battery and a transmission coil to travel around the sensing field and charge the sensors wirelessly. Here, the sensors have a receiver that helps them receives energy from the MC through electromagnetic waves. Unlike the energy harvesting techniques, this scheme offers agile, controllable, and reliable energy replenishment, thus enabling genuinely sustainable operations for the sensor networks.

In practice, the charging strategy of the MC(s) has a considerable impact on the performance of a WRSN (i.e., sensor lifetime). Numerous research has been conducted to design charging strategies for the MC, which can be classified into two categories: *periodic* charging and *on-demand* charging. In the periodic charging scheme, the energy consumption rate of each sensor is assumed to be constant and known in advance. Thereby, an optimal charging trajectory could be planned before the running phase. The MC then travels along the pre-optimized charging trajectory to recharge nodes in a periodic and deterministic manner (Jiang et al., 2017, Lyu et al., 2019, Ma et al., 2018b, Xu et al., 2019). However, the sensors' energy consumption profiles often fluctuate greatly due to the close interaction with the surrounding environment (He et al., 2013). Furthermore, WSNs are very dynamic–that a node failure might change the routing paths of many packets, causing turbulence in the average power consumption of many nodes. As a result, the pre-optimized charging trajectory may become inefficient.

To overcome these issues, the on-demand charging scheme instead requires the sensors to send a charging request to the MC when their residual energy falls below a predetermined threshold. The MC will maintain a pool of charging requests and then choose the next charging destinations based on the current service pool. He et al. (He et al., 2013) introduced a simple heuristic algorithm, namely Nearest-Job-Next with Preemption (NJNP), that chooses the next charging destination according to the spatially closest sensor in the queue. Several following works (Fu et al., 2015, Kaswan et al., 2018, Lin et al., 2017, 2019, Zhu et al., 2018) improved this heuristic by introducing double warning thresholds or using meta-heuristic algorithms. Recent works also leveraged (deep) Q-learning to learn on-demand charging policy (Cao et al., 2021, La et al., 2020).

Despite the promising results, a common drawback of the on-demand algorithms is the dependence on the chosen threshold for the charging requests. Specifically, if the charging threshold is set too high, the sensors will send the charging requests too frequently, causing the MC to become overloaded and degrading the efficiency of the charging algorithms (Zhu et al., 2018). A small charging threshold, on the other hand, may cause sensors to submit requests too late, and the MC may not come in time to charge before the sensor runs out of energy. In the example shown in Fig. 5.1.1, after fully charging node F, the mobile charger (MC) continuously chooses a requested node in the current service

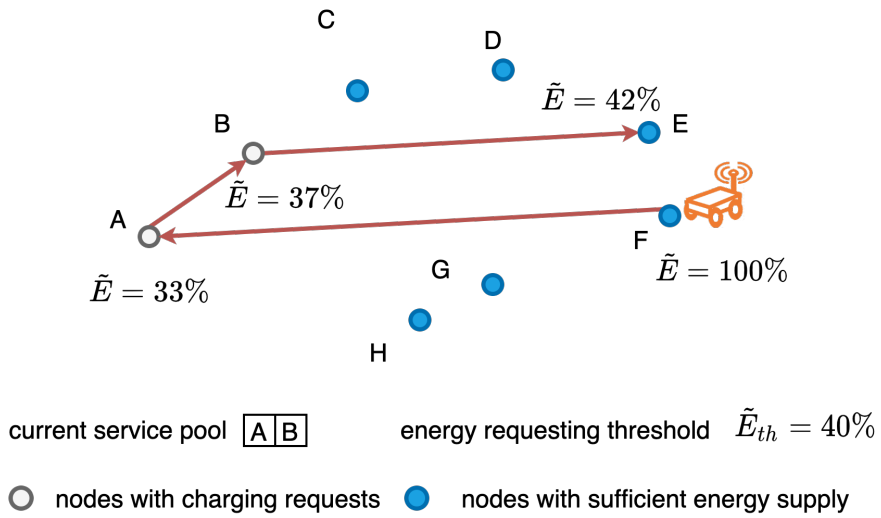


Figure 5.1.1: The drawback of on-demand charging scheme. Node E sends a charging request right after the MC decides to charge node A next.

pool to charge (node A or B). If node E requests charging when the MC has just decided to go to A or B, the MC must move back and forth to serve the requests.

In this work, we propose a novel scheme called *adaptive charging* to address the drawbacks of the existing approaches. Specifically, we eliminate the energy threshold in the on-demand charging and endow the MC with the ability to choose any sensor to charge at any time, depending on the current state of the network. Thus, the charging policy is a mapping between the network's state space and all possible actions, specifying what to do at a given state. We leverage a deep neural network (DNN) to approximate the strategy's mapping and use reinforcement learning (RL) techniques to train the model. The MC will act as an agent in the sensor network (environment), learning to derive the optimum charging decisions by interacting with the environment. We will train the model with considerable network topologies in advance; thus, the trained model can be applied to any network topology without manually adjusting the parameters.

Contributions. We consider here the connected target coverage problem (Zhao and Gurusamy, 2008), under the setting of the wireless rechargeable sensor network (WRSN) paradigm. Specifically, several critical targets are required to be continuously monitored by a number of randomly scattered sensors, which can be replenished wirelessly by an MC. Our objective is to design a charging strategy for the MC to maximize the time interval that all targets are continuously monitored and analyzed by the BS. The main contributions of this paper are as follows.

- We propose an adaptive charging scheme by omitting the energy threshold for charging requests in the on-demand scheme and endowing the MC ability to charge any sensor at any time. This enables the MC to evaluate other charge options that may improve the cumulative network's lifespan instead of focusing on the service pool.
- We design a charging policy to prolong the network's lifetime using a DNN model, which can be trained by RL techniques. Our model is flexible–it can operate in a dynamic sensor network where the number of sensors might change due to node failures or deployments.
- We conduct the experiment showing the superiority of our model compared to existing on-demand methods in prolonging the network lifetime. We then discuss further

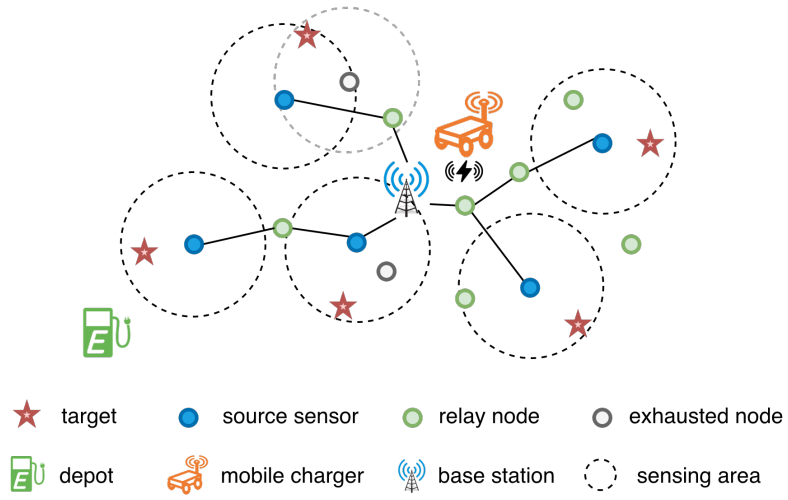


Figure 5.2.1: An illustration of a wireless rechargeable sensor network for target-covering.

the self-organizing capability of the proposed method.

The rest of the paper is unfolded as follows: In the next section, we describe the system model and the setup of the connected target coverage problem in WRSNs. Section 5.3 presents our learning model for the adaptive charging strategy of the MC. In Section 5.4, we conduct experiments to demonstrate the efficiency of the proposed method.

5.2 System Model and Problem Statement

5.2.1 System Model

Figure 5.2.1 depicts our targeted wireless rechargeable sensor network, which comprises four main components: sensors, a base station, a depot, and a mobile charger. Each sensor has a sensing unit, a processing unit, a transceiver unit, and a power unit. The sensors monitor pre-specified targets and transmit sensory data to the base station in a multi-hop paradigm. A mobile charger is a device equipped with a wireless energy transfer module. It will travel around the network and wirelessly transmit power to the sensors. When the mobile charger's energy is almost exhausted, it will return to the depot to charge.

Each sensor has a sensing area determined by its location and sensing range. Any target located in the sensing area of a sensor could be monitored. A sensor is called a *source node* if it covers at least one target, while the sensors that do not cover any target may act as relay nodes forwarding the sensing data to the sink. A source sensor will perform the monitoring task and periodically generate sensing data. Here, we assume that all source sensors generate sensing data at the same rate, i.e., all sensors have the same sampling frequency, quantization, and coding scheme (Zhao and Gurusamy, 2008). Thus, each source sensor generates a fixed amount of bits per unit of time. The sensing data gathered by source sensors is transmitted to the sink by multi-hop communication over other sensors. Two sensors can communicate with each other if the distance between them is less than the communication range. We use the same energy consumption model as in Section 4.2.2.

5.2.2 Charging Model

In this work, we only allow the MC to charge one sensor at a time and the sensor will be fully charged. A charging action of the MC consists of two phases: (1) moving into the vicinity of the sensor and (2) charging the sensor to full battery capacity. The time to charge a sensor is determined as follows:

$$t_i^{\mathrm{ch}} = \frac{B^{\mathrm{SN}} - e_i}{\mu - \omega_i}, \tag{5.1}$$

where μ is the charging rate, B^{SN} and e_i denote the battery capacity and the residual energy of the sensor, respectively. The parameter ω_i is the energy consumption rate (ECR) of the sensor i which will be estimated by the BS using the residual energy profile of the sensor. The residual energy profile of the sensor is the sequence of notification packets, i.e., the sensor records its residual energy and current time stamp and periodically sends it to the BS. This procedure is similar to the estimation of the dynamic energy consumption rate in (Zhu et al., 2018).

The dissipated energy of the MC for traveling l units of distance and charging the sensor i is calculated as:

$$\tilde{E}_{\mathrm{MC}}(l, e_i, \omega_{\mathrm{move}}) = l\omega_{\mathrm{move}} + \mu t_i^{\mathrm{ch}}. \tag{5.2}$$

Here, similar to Cao et al. (2021), we omit the energy dissipated into the environment during charging; thus the energy consumption of the MC is only composed of the energy transferred to sensor nodes and the energy consumed by traveling.

5.2.3 Problem Statement

Let us denote by $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ the set of n deployed sensors and by $\mathcal{Q} = \{q_1, q_2, \ldots, q_m\}$ the set of m targets to be monitored. We also refer to p^{BS} and p^{D} as the position of the BS and the depot, respectively. All nodes are deployed in a two-dimensional sensing area. All sensors have the same battery capacity B^{SN}, sensing range r_s, and communication range r_c. A target q_k can be monitored by a sensor p_i if $\mathrm{dist}(p_i, q_k) \leq r_s$, where $\mathrm{dist}(\cdot, \cdot)$ denotes the Euclidean distance. Meanwhile, two sensors p_i and p_j can communicate to each other if $\mathrm{dist}(p_i, p_j) \leq r_c$. Initially, we assume that each target is covered by at least one sensor and that at least one transmission route from each sensor toward the BS exists. When a sensor depletes its energy, it deactivates itself and waits to be replenished.

A network's state is considered covered and connected if it satisfies the two following constraints: (1) *coverage*, each target is covered by at least one source sensor, (2) *connectivity*, from each source sensor to sink, there must exist at least one route traversing through only active sensors. We then define the *network lifetime* as the time interval from when the network starts till the target coverage or the connectivity is not satisfied.

The MC's parameters can be represented by four factors $\langle B^{\mathrm{MC}}, \nu, \mu, \omega_{\mathrm{move}} \rangle$, where B^{MC} is the battery capacity, ν is the traveling speed, ω_{move} denotes the energy consumption rate of the MC per one unit distance, and μ accounts for both charging rates from the MC to a sensor and from the depot to the MC. For the sake of simplicity, we assume that the velocity of the MC is constant and the MC can travel freely inside the sensor field without obstacles.

The charging path of the MC is a sequence of charging locations which are the positions

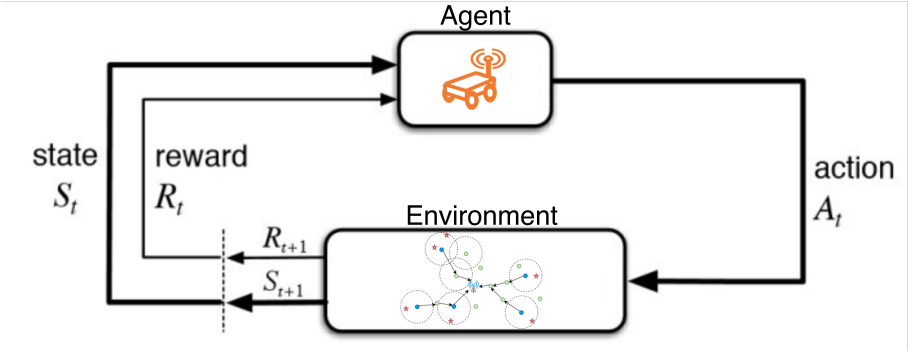


Figure 5.3.1: Learning model of a reinforcement learning system.

of the sensors or the depot. If the MC travels to a sensor, the MC will charge it to full capacity. If the MC goes back to the depot, the MC will recharge its own battery fully. Our objective is to determine a charging path that maximizes the lifetime of the sensor network so that every target is monitored and analyzed continuously by the BS.

Notice that our scheme does not require the sensors to send the charging requests to the MC. We endow the MC the ability to choose any charging destination at any time based on the network's state and its own status. Thus we do not need to predetermine the energy threshold for charging requests like the existing on-demand charging schemes (Cao et al., 2021, He et al., 2013, Zhu et al., 2018).

5.3 Proposed Method

The goal of this paper is to design a charging strategy for the MC that can adapt to the uncertainties arising in the sensor network. To this end, we propose a novel adaptive charging scheme based on deep reinforcement learning. The MC will act as an intelligent agent interacting with the sensor network by moving around to replenish sensors or itself (Fig. 5.3.1). We model the policy using a deep neural network that takes a network's state as the input and outputs the probabilities of taking charging decisions. The model is then trained by interacting with simulated environments and adjusting the policy to the return from the environments.

5.3.1 Formulation of the DRL Framework

In the following, we formulate our problem under the DRL framework. The mathematical model of a DRL model typically consists of four items, namely a state space $\mathcal{S}$, an actions space $\mathcal{A}$, a transition model T, and a reward function R.

State. The state information represents the status of the network and the MC that helps the MC choose the next charging locations. We divide the useful information into two groups: *static* and *dynamic* elements. The static elements contain prescribed information related to the properties of the mobile charger, the depot, or the sensors such as the position of sensors, battery capacity, and the number of targets covered by a sensor. The dynamic elements include temporal information such as the residual energy of devices, the current position of the MC, and the estimated energy consumption rate of each sensor. Formally, we define a state $x \in \mathcal{S}$ as a tuple of $(x^{MC}, x^D, \bar{x}^{SN})$, where x^{MC} is a tuple of the static and dynamic information of the mobile charger, x^D contains the location of the depot in the sensor field, and $\bar{x}^{SN} = \{x_i^{SN}, i = 1, \ldots, n\}$ is a sequence of tuples containing static

Table 5.3.1: State information. The notation S and D indicate static and dynamic information, respectively.

State	Parameter	Type	Comment
x^{MC}	B^{MC}	S	battery capacity of the MC
	ν	S	velocity of the MC
	μ	S	charging rate
	ω_{move}	S	ECR for traveling
	p^{MC}	D	current position
	e^{MC}	D	residual energy power
x_i^{SN}	B^{SN}	S	battery capacity of a sensor
	p_i	S	position of sensor i
	ξ_i	S	no. of targets covered by sensor i
	e_i	D	residual energy of sensor i
	ω_i	D	ECR of sensor i
x^D	p^{D}	S	position of the depot

and dynamic information of each sensor. The detail of the state information is shown in Table 5.3.1.

Action. We define $n+1$ actions corresponding to $n+1$ charging destinations (a depot and n sensors). An action a_t made at time t is an integer number $a_t \in \mathcal{A} = \{0, 1, 2, ..., n\}$, where we denote $a_t = i, i > 0$ with regard to the integral index of the sensors, and $a_t = 0$ corresponds to going back to the depot and recharging itself.

Transition. We simulate the network environment and return the network's state at the end of doing the action a_t as the next state s_{t+1}. The dissipated energy of the sensors and the MC will be computed following Eq. (4.3) and (5.2). It is worth noting that the MC is only able to observe the returned state s_{t+1}, not the underlying process of the simulation. Compared to Bui (2021), we add some random noise to the resulting states of the environment after each action to simulate the uncertainties of the real world environment.

Reward. We define the reward $R(x_t, a_t)$ of doing an action a_t at state x_t as the period of time doing the action which includes both traveling and charging time. If either the coverage or connectivity conditions are not met, the simulation environment will be terminated immediately and return the reward till the network downtime. Thus, cumulative reward over all actions coincides with the lifetime of the network.

A charging trajectory can be represented as a sequence of the network's states and the MC's actions $\tau = \{x_0, a_0, \ldots, x_{T-1}, a_{T-1}, x_T\}$, $a_t \in \mathcal{A}$, $x_t \in \mathcal{S}$. The γ-discounted lifetime of the trajectory τ can be computed as:

$$G(\tau) = \sum_{t=0}^{T} \gamma^t R(x_t, a_t). \tag{5.3}$$

A stochastic policy $\pi(a|x)$ determines the probability of taking charging action a given network state x which models the MC's behavior at a given time. We aim to find a policy π^* that maximizes the expected γ-discounted network's lifetime.

We delineate a policy approximation using a deep neural network and a policy gradient algorithm to train the model in the following sections.

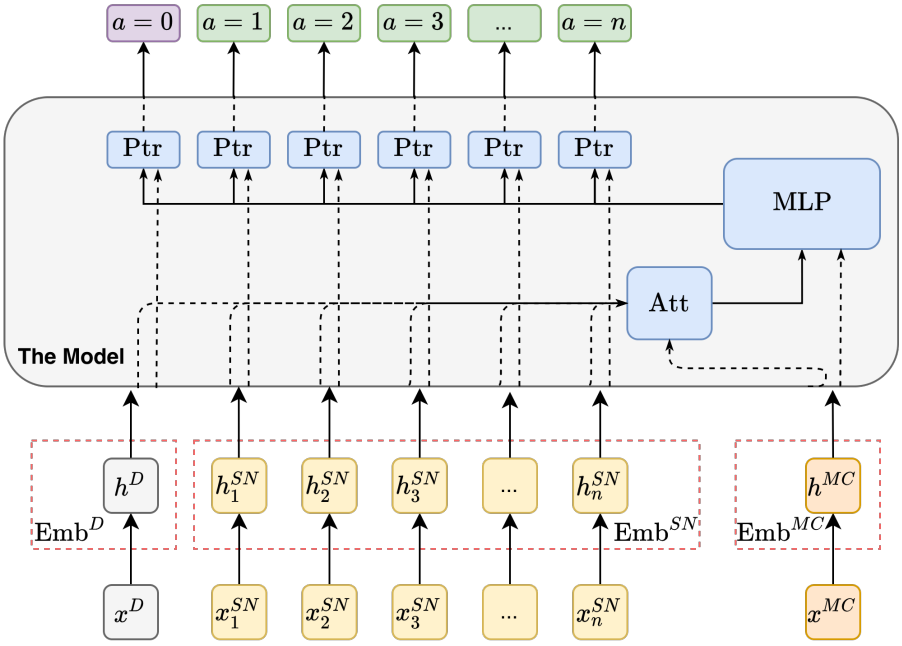


Figure 5.3.2: The model architecture of the actor.

5.3.2 Model Architecture

We parameterize the stochastic policy by $\pi_\theta(a|x)$, where θ is the parameters of a deep neural network aided by the attention and pointing mechanisms, similar to (Hottung and Tierney, 2019). Fig. 5.3.2 depicts the overall architecture.

The input to the model is the network state x_t at time t which is composed of the states at time t of the MC ${x^{MC}}_t$, the depot x^D, and the sensors $\bar{x}_t^{SN}$. The output of the model is the probabilities of taking action a_t given the state x_t. To simplify the notation, in this section, we omit the timestamp t when referring to the states x_t, ${x^{MC}}_t$, and $\bar{x}_t^{SN}$.

We use three transformations Emb^{MC}, Emb^{D}, and Emb^{SN} to embed the input of the mobile charger, the depot, and the sensors, respectively, to a d-dimensional latent space. Note that we use the same transformation Emb^{SN} for all sensors and the embedding vector of each sensor is computed separately and identically. Let $h^{MC}, h^D, h_i^{SN} \in \mathbb{R}^d$ be the embedded input corresponding to x^{MC}, x^D, x_i^{SN}. For convenience's sake, we denote $\bar{h}^C = \{h^D, h_1^{SN}, ...h_n^{SN}\}$ as a matrix of embedded input of charging destinations and then refer to $\bar{h}_i^C$ as the embedding of charging destination i. An attention layer is used to extract alignment vector $\bar{a}$, which specifies how much 'attention' the MC might have for each charging destination given their current status. Precisely, the alignment vector is calculated by the following formula:

$$\bar{a} = \text{softmax}(u_0^H, u_1^H, ..., u_n^H), \tag{5.4}$$

where:

$$u_i^H = z^A \tanh\left(W^A[\bar{h}_i^C; h^{MC}]\right). \tag{5.5}$$

Here, $[;]$ denotes the concatenation of two vectors. The context vector c is provided by:

$$c = \sum_{i=0}^{n} \bar{a}_i \bar{h}_i^C. \tag{5.6}$$

The context vector is later concatenated with the embedded input of the MC to be the input of a multilayer perceptron (MLP) with one hidden layer that outputs a vector $q \in \mathbb{R}^d$.

$$q = \mathrm{MLP}_{W^B}([c; h^{MC}]). \tag{5.7}$$

The distribution of the policy over all actions for the state x is then given by:

$$\pi_\theta(a = i|x) = \mathrm{softmax}(u_0, u_1, ..., u_n), \tag{5.8}$$

where

$$u_i = z^C \tanh(\bar{h}_i^C + q), \tag{5.9}$$

and $\theta = \{z^A, W^A, W^B, z^C\}$ are trainable parameters.

The pointing mechanism (Vinyals et al., 2015) in the Eq. (5.8) is a reduction of attention mechanism that leverages the alignment vector to determine the probabilities of selecting each member in the input. Both only require learning a tuple of parameters to compute utilities of charging actions ((z^A, W^A) for attention and z^C for pointing). It thus imposes the permutation invariant property for the set of the input's members, i.e., the output probabilities of the policy depend solely on the features of the sensors but not the order of the sensors in the input state. Additionally, it allows the MC to operate on a dynamic network where the number of sensors might change due to node failures or node deployments. It differs from the prior DRL work for the on-demand approach (Cao et al., 2021), which requires modifying the architecture and retraining the model when the number of sensors is changed. Moreover, instead of using a Gated Recurrent Unit (GRU) as in the pointer network (Vinyals et al., 2015), we use a fully connected MLP, similar to (Hottung and Tierney, 2019), to avoid the dependence of the MC's decision on the previous state. It enables the trained MC to be deployed on the fly into an existing WSN.

5.3.3 Policy Optimization

We train the agent using a well-known policy gradient method in reinforcement learning. Our objective is to maximize the expected network's lifetime:

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t R(x_t, a_t) \right], \tag{5.10}$$

where $p_\theta(\tau)$ is the distribution of the Markov chain induced by the policy π_θ which generates the trajectories τ. Applying the REINFORCE (Williams, 1992), the gradient of Eq. (5.10) can be computed as:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \nabla_\theta \log(\pi_\theta(a_t|x_t)) A_t \right], \tag{5.11}$$

where $A_t = \sum_{l=t}^{T} \gamma^{l-t} R(x_t, a_t) - V_\psi(x_t)$ is the advantage function of taking action a_t given state x_t. This vanilla policy gradient update has no bias but high variance, which leads to an unstable learning process. To overcome this issue, we use the Generalized Advantage Estimation (GAE) (Schulman et al., 2015) to reduce the variance caused by the original advantage function at the cost of introducing bias. Furthermore, due to the non-convex objective function, the policy gradient usually suffers from local convergence.

Algorithm 5 Policy gradient algorithm

Input: A set of network instances $\mathcal{D}$, discount factor γ, GAE hyperparameter λ, regularization hyperparameter β.

Output: A trained policy π_θ.

initialize the actor network with random weight θ.
initialize the critic network with random weight ψ.
for $\text{epoch} = 1, 2, ...$ **do**
  for $n = 1, ..., N$ **do**
    initialize the environment on the instance $D_n \in \mathcal{D}$.
    generate an episode following π_θ:
      $x_0, a_0, x_1, a_1, ..., x_{T-1}, a_{T-1}, x_T$
    $G \leftarrow 0, A_t^{\text{GAE}} \leftarrow 0$
    $d\theta \leftarrow 0, d\psi \leftarrow 0$
    for $t = T-1, T-2, ..., 0$ **do**
      $G \leftarrow \gamma G + R_t$
      $\delta_t \leftarrow R_t + \gamma V_\psi(x_{t+1} - V_\psi(x_t)$
      $A_t^{\text{GAE}} \leftarrow \gamma\lambda A_t^{\text{GAE}} + \delta_t$
      $\mathrm{d}\theta \leftarrow d\theta - \nabla_\theta \log(\pi_\theta(a_t|x_t) A_t^{\text{GAE}} - \beta\nabla_\theta H(\pi_\theta(\cdot|x_t))$
      $d\psi \leftarrow d\psi + \nabla_\psi \frac{1}{2} \|G - V_\psi(x_t)\|_2^2$
    end for
    $\theta \leftarrow \text{Adam}(\theta, d\theta)$
    $\psi \leftarrow \text{Adam}(\psi, d\psi)$
  end for
end for

We hence use the entropy regularization as suggested in (Mnih et al., 2016) to encourage the exploration. The following formula computes the final policy gradient:

$$\nabla J(\theta) = \mathbb{E}_{\tau\sim p_\theta(\tau)}\Bigg[\sum_{t=0}^{T} \nabla_\theta \log\left(\pi_\theta(a_t|x_t)\right)\hat{A}_t^{\text{GAE}(\lambda)} + \beta\nabla_\theta H(\pi_\theta(\cdot|x_t))\Bigg], \tag{5.12}$$

where H is the entropy function, β is a hyperparameter controlling the strength of the regularization, and $\hat{A}_t^{\text{GAE}(\lambda)}$ is the GAE function, which is estimated by:

$$\hat{A}_t^{\text{GAE}(\lambda)} = \sum_{l=t}^{T} (\gamma\lambda)^l (R_l + \gamma V_\psi(x_{l+1}) - V_\psi(x_l)), \tag{5.13}$$

where λ is a hyperparameter controlling the trade-off between variance and bias in the advantage function, $V_\psi(x)$ is the estimated value function at the state x.

During training, we maintain two networks, one for the policy $\pi_\theta(a|x)$, and another for value function $V_\psi(x)$ with the trainable parameter θ and ψ, respectively. We use a simple three-layer fully connected MLP with a ReLU activation in between for the value function's network. The policy is updated using gradients computed in Eq. (5.12). Meanwhile, we use Mean Square Error (MSE) to compute the loss function for the value function. Both networks are trained with *Adam* optimizer (Kingma and Ba, 2014). The pseudocode

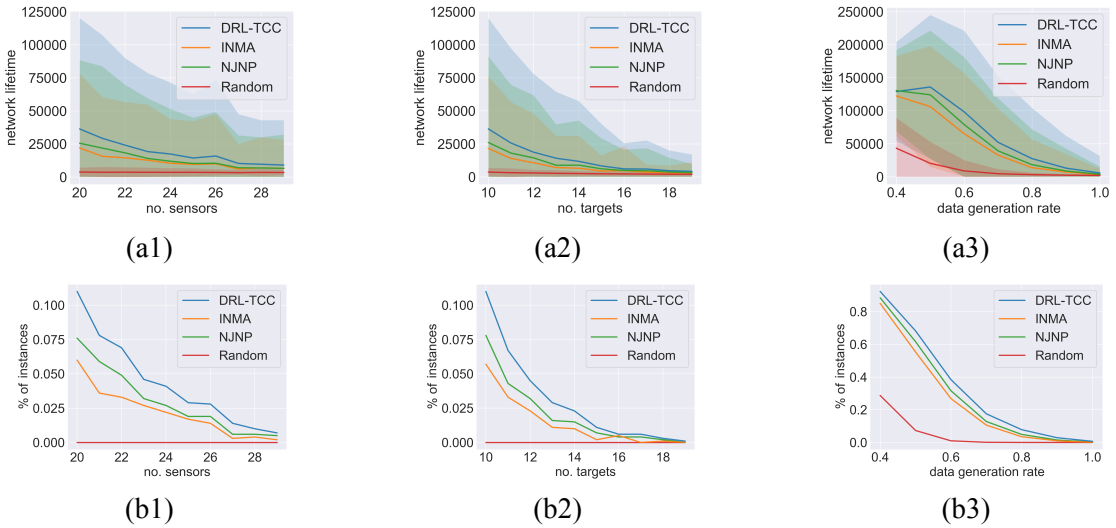


Figure 5.3.3: Evaluation of the network lifetime of competing algorithms when varying the number of sensors, number of targets, or package generation probability.

of our training process is described in Algorithm 5.

5.4 Experiments

In this section, we conduct simulations to demonstrate the merits of our method compared to existing on-demand charging algorithms. We also discuss the self-organizing capability of the wireless sensor networks powered by our mobile charger. The model and the environment simulation are implemented using the PyTorch and Gym framework. All experiments are performed in a computer with an Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz, 16 GB RAM, and GeForce GTX TITAN X GPU running on Ubuntu Linux 18.04.

5.4.1 Simulation Settings

Network structure. We assume that the sensor network is randomly deployed in a square area of interest 200×200. The sink is supposed to be located in the center $(100, 100)$ of the field, and the depot is in the bottom-left corner $(0, 0)$. Initially, all sensors are equipped with a full and rechargeable battery with the capacity of $B^{\mathrm{SN}} = 10J$. Meanwhile, the MC, which has a high-capacity battery ($B^{\mathrm{MC}} = 500J$), is initially located at the depot with an initial energy of $50J$. The reason for the low initial power setting is to enforce the exhausted state of the MC in some first states, which encourages the MC to learn when to go back depot for recharging in some first episodes. That, in turn, accelerates the learning process. We set the other parameters as in Table 5.4.1, similar to (He et al., 2013).

Implementation. To train a DRL model, we generate 10000 network instances with 20 sensors and 10 targets. The positions of sensors and targets are drawn in a square area according to the uniform distribution. We assume that all instances are guaranteed to be covered and connected initially. The probability of a sensor sending a packet of sensing data in a unit time period κ (data generation rate) is set at 0.8. For the training parameters, we set the reward discount γ, the entropy regularization β, and GAE λ to 0.95, 0.02, and 0.9, respectively. We then train one DRL model in these environments and fix the agent throughout the evaluations.

Table 5.4.1: Configuration of the simulations.

Parameter	Value	Unit	Comment
n	$20 \sim 30$	–	number of deployed sensors
m	$10 \sim 20$	–	number of critical targets
B^{MC}	500	J	battery capacity of the MC
ω_{move}	0.04	J/m	ECR of the MC for traveling
ν	5	m/s	velocity of the MC
B^{SN}	10	J	battery capacity of a sensor
r_s	40	m	sensing range
r_c	80	m	communication range
μ	0.04	J/s	charging rate

Baselines. We mainly compare our method, namely DRL-TCC, against the existing on-demand charging algorithms:

- Random: The agent chooses the next charging destination at random. We add a simple estimation that helps the agent go back to the depot to recharge itself before being exhausted.
- NJNP (He et al., 2013): A heuristic algorithm with a simple but very efficient discipline that chooses the spatially closest requesting node as the next charging destination.
- INMA (Zhu et al., 2018): A modified algorithm of NJNP that selects nodes that make the least number of other requesting nodes enduring energy deficiency as the charging candidates. For high charging efficiency, the node with the shortest time to finish the charging will be selected as the next charging node if the candidate set has more than one node.

5.4.2 Performance Evaluation

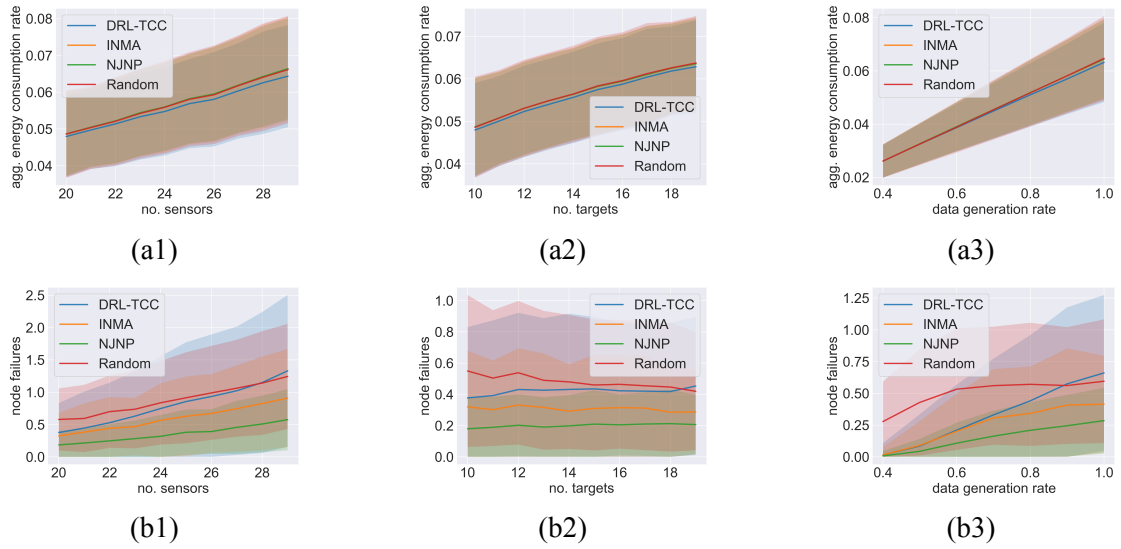


Figure 5.4.1: The comparison of the aggregated energy consumption rate and the number of node failures when increasing the number of sensors, number of targets, or data generation rate.

We assess the ability to prolong the network's lifetime of the evaluated algorithms on various topologies. To understand the impact of the number of sensors (n), the number

of targets (m), and the data generation rate (κ) on the performance of the competing algorithms, we vary one factor while leaving the other two as the default values ($n = 20$, $m = 10$, $\kappa = 0.8$). For each configuration, we generate 1000 network instances where the sensors' and targets' positions are drawn from the same uniform distribution. We then simulate the WRSN operations and report the average cumulative lifetime over all instances.

To prevent endless simulation, for each instance, we limit the maximum charging actions of the mobile charger to 2000. We then report the number of instances with respect to which the system still sustains the full coverage and connectivity after having done the charging limit. The results are shown in Fig. 5.3.3.

Impact of n, m. We vary the number of sensors from 20 to 30 (Figure 5.3.3a1 and 5.3.3a2), and the number of targets from 10 to 20 (Figure 5.3.3b1 and 5.3.3b2). Generally, the network lifetime decreases when increasing the number of sensors or targets since deploying more sensors or targets leads to an increase in network load and MC's burden. The high variance can be observed in all settings that demonstrate the dependence on the network topology of the charging algorithms. However, our method dominates INMA, NJNP, and Random in all settings. When $n = 20$, the DRL-TCC extends the network lifetime to around $36375s$ on average of 1000 network instances. Meanwhile, the MC running with NJNP, INMA, and Random strategy, only conserves the network to around $25600s$, $22027s$, and $3780s$, respectively. Moreover, there are 110 out of 1000 network instances at which the MC with DRL-TCC elongates the network to over 2000 charging actions. The numbers of NJNP, INMA, and Random are 76, 60, and 0, respectively.

Impact of κ. We vary the data generation rate κ from 0.4 to 1.0 and report the comparison in Fig. 5.3.3a3 and 5.3.3b3. Similar to the experiments with sensors and targets, the average lifetime decreases as the data generation rate increases. However, DRL-TCC still shows superior results compared to NJNP and INMA. Specifically, the result of DRL-TCC outperforms 30.57% on average compared to the result given by NJNP and 57.33% better than the result of INMA. Random continues to show the worst results.

Self-organizing capability. To further probe the self-organizing capability of the proposed method, we report the aggregated energy consumption rate and the average number of node failures associated with the aforementioned experiments. The former is the average amount of energy consumed by the sensors in a unit of time and the latter is the average number of exhausted nodes. We report the average over all instances for both metrics in Fig. 5.4.1.

We can observe that the aggregated energy consumption rate of DRL-TCC is slightly lower than others, while the number of node failures is higher than INMA and NJNP when increasing the network load. Coupled with the superiority in the performance of prolonging the network's lifetime, it suggests that the DRL-TCC decided to leave *non-critical* nodes to be exhausted to reduce the network load. The transmission topology will be reorganized with a lower network load. This enables the MC to alleviate its burden; it thus elongates the network's lifetime while still ensuring coverage and connectivity. Notice that, despite showing higher performance in the network's lifetime than the Random strategy, the aggregated energy consumption rate of INMA and NJNP has no significant difference compared to Random.

5.5 Discussion

This paper studied the connected target coverage problem under the settings of the WRSN paradigm. We proposed a novel scheme for scheduling the charging path for the MC to prolong the network lifetime. Our scheme relies on deep reinforcement learning techniques to design a charging policy that takes the network state as the input and outputs the probability of each charging action. Our model is a combination of attention and pointing mechanisms that facilitate the operation of a dynamic network where the number of sensors might be changed due to node failures or deployments. The empirical results show the superiority of our method compared to the existing on-demand algorithms.

One limitation is that the simulation scenario is simplified. The authors did not consider many practical issues that could occur in a real-world deployment, such as sensor mobility, variations in sensor energy consumption, and the presence of obstacles in the environment. These factors could affect the effectiveness of the proposed scheme in extending network lifetime, and it would be interesting to investigate how the proposed approach performs under more complex and realistic scenarios.

Another limitation is that the proposed scheme relies on a deep reinforcement learning approach, which requires a large amount of data to train the model. This could be a challenge in a real-world scenario where data collection might be limited or costly. Moreover, the proposed scheme may be sensitive to the quality and quantity of the training data, and it is unclear how the approach would perform with limited training data. These could be potential directions for future research.

Conclusion and Future Work

This thesis studied two emerging problems to prolong the lifetime of WSNs: energy-efficient routing via relay node placement and adaptive charging schemes in WRSNs. For each problem, we proposed a novel idea to enhance the efficiency of the wireless network while ensuring its connectivity and reliability in harsh environments.

Specifically, for the relay node placement problem, we focused on a multi-objective setting in which the minimum additional RNs are considered for efficient deployment cost, and balances in energy consumption are considered to prolong the network lifetime. Additionally, we introduced a hop count bound as a network delay surrogate constraint for network reliability. To solve our problem, we proposed a multi-objective evolutionary algorithm with novel objective-oriented heuristics that aid crossover and mutation operators for better convergence speed. Extensive evaluations with 3D-terrain simulations showed that our framework has a better trade-off between objectives than existing algorithms.

For the second problem, we studied the connected target coverage problem under the settings of the WRSN paradigm. We proposed a novel scheme for scheduling the charging path for the MC to prolong the network lifetime. Our scheme relies on deep reinforcement learning techniques to design a charging policy that takes the network state as the input and outputs the probability of each charging action. Our model is a combination of attention and pointing mechanisms that facilitate the operation of a dynamic network where the number of sensors might be changed due to node failures or deployments. The empirical results show the superiority of our method compared to existing algorithms.

In the future, integrating multiple techniques could be a focus to prolong the network lifetime while ensuring network coverage, connectivity, and data quality. It would be valuable to analyze the impact of routing strategy on wireless charging, which could facilitate the integration of these two techniques. Additionally, researchers could investigate the practicality and cost-effectiveness of implementing these techniques in real-world scenarios. Integrating these techniques could lead to a more robust and sustainable WSN, which could be utilized in diverse applications, such as environmental monitoring, smart cities, and precision agriculture. The potential benefits of integrating these techniques underscore the importance of continued research in this area.

Related Papers

- Ngoc Bui, Phi Le Nguyen, Viet Anh Nguyen, and Phan Thuan Do. A deep reinforcement learning-based adaptive charging policy for wrsns. In *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 661–667. IEEE, 2022
- Ngoc Bui, Tam Nguyen, Huynh Thanh Binh, and Le Trong Vinh. A phenotype-based multi-objective evolutionary algorithm for maximizing lifetime in wireless sensor networks with bounded hop. *under review at Soft Computing*, 2023

Bibliography

Mohammed Abdulkarem, Khairulmizam Samsudin, Fakhrul Zaman Rokhani, and Mohd Fadlee A Rasid. Wireless sensor network for structural health monitoring: a contemporary review of technologies, challenges, and future direction. *Structural Health Monitoring*, 19(3):693–735, 2020.

Kofi Sarpong Adu-Manu, Nadir Adam, Cristiano Tapparello, Hoda Ayatollahi, and Wendi Heinzelman. Energy-harvesting wireless sensor networks (eh-wsns) a review. *ACM Transactions on Sensor Networks (TOSN)*, 14(2):1–50, 2018.

Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

A Alphonsa and G Ravi. Earthquake early warning system by iot using wireless sensor networks. In *2016 International conference on wireless communications, signal processing and networking (WiSPNET)*, pages 1201–1205. IEEE, 2016.

Charles Audet, Jean Bigeon, Dominique Cartier, Sébastien Le Digabel, and Ludovic Salomon. Performance indicators in multiobjective optimization. *European journal of operational research*, 2020.

Miloud Bagaa, Ali Chelli, Djamel Djenouri, Tarik Taleb, Ilangko Balasingham, and Kimmo Kansanen. Optimal placement of relay nodes over limited positions in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 16(4):2205–2219, 2017.

Yosra Zguira Bahri. *Study and development of wireless sensor network architecture tolerant to delays*. PhD thesis, Université de Lyon; Université du Centre (Sousse, Tunisie), 2018.

Trupti Mayee Behera, Umesh Chandra Samal, Sushanta Kumar Mohapatra, Mohammad S Khan, Bhargav Appasani, Nicu Bizon, and Phatiphat Thounthong. Energy-efficient routing protocols for wireless sensor networks: Architectures, strategies, and performance. *Electronics*, 11(15):2282, 2022.

Abhijit Bhattacharya and Anurag Kumar. A shortest path tree based algorithm for relay placement in a wireless sensor network and its performance analysis. *Computer Networks*, 71:48–62, 2014.

Ngoc Bui. A python framework for genetic-based algorithms, October 2020.

Ngoc Bui. *A Deep Reinforcement Learning-based Online Charging Scheme for Target Coverage and Connectivity in WRSNs*. PhD thesis, Hanoi University of Science and Technology, 2021.

Ngoc Bui, Phi Le Nguyen, Viet Anh Nguyen, and Phan Thuan Do. A deep reinforcement learning-based adaptive charging policy for wrsns. In *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 661–667. IEEE, 2022.

Ngoc Bui, Tam Nguyen, Huynh Thanh Binh, and Le Trong Vinh. A phenotype-based multi-objective evolutionary algorithm for maximizing lifetime in wireless sensor networks with bounded hop. *under review at Soft Computing*, 2023.

Xianbo Cao, Wenzheng Xu, Xuxun Liu, Jian Peng, and Tang Liu. A deep reinforcement learning-based on-demand charging algorithm for wireless rechargeable sensor networks. *Ad Hoc Networks*, 110:102278, 2021.

Ran Cheng, Yaochu Jin, Kaname Narukawa, and Bernhard Sendhoff. A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. *IEEE Transactions on Evolutionary Computation*, 19(6):838–856, 2015.

Carlos A Coello Coello and Nareli Cruz Cortés. Solving multiobjective optimization problems using an artificial immune system. *Genetic programming and evolvable machines*, 6(2):163–190, 2005.

Balázs Csanád Csáji, Zsolt Kemény, Gianfranco Pedone, András Kuti, and József Váncza. Wireless multi-sensor networks for smart cities: A prototype system with statistical data analysis. *IEEE Sensors Journal*, 17(23):7667–7676, 2017.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Mohammed Farsi, Mostafa A Elhosseini, Mahmoud Badawy, Hesham Arafat Ali, and Hanaa Zain Eldin. Deployment techniques in wireless sensor networks, coverage and connectivity: A survey. *Ieee Access*, 7:28940–28954, 2019.

Igor Florinsky. *Digital terrain analysis in soil science and geology*. Academic Press, 2016.

Lingkun Fu, Liang He, Peng Cheng, Yu Gu, Jianping Pan, and Jiming Chen. Esync: Energy synchronized mobile charging in rechargeable wireless sensor networks. *IEEE Transactions on vehicular technology*, 65(9):7415–7431, 2015.

Gordana Gardašević, Konstantinos Katzis, Dragana Bajić, and Lazar Berbakov. Emerging wireless sensor networks and internet of things technologies—foundations of smart healthcare. *Sensors*, 20(13):3619, 2020.

Rohit D. Gawade and S. L. Nalbalwar. A centralized energy efficient distance based routing protocol for wireless sensor networks. *Journal of Sensors*, 2016, 2016. ISSN 16877268. doi: 10.1155/2016/8313986.

Jens Gottlieb, Bryant A Julstrom, Günther R Raidl, and Franz Rothlauf. Prüfer numbers: A poor representation of spanning trees for evolutionary search. In *Proceedings of the genetic and evolutionary computation conference*, volume 343, page 350. Citeseer, 2001.

Nitin Goyal, Mayank Dave, and Anil K Verma. Data aggregation in underwater wireless sensor network: Recent approaches and issues. *Journal of King Saud University-Computer and Information Sciences*, 31(3):275–286, 2019.

Kalpna Guleria and Anil Kumar Verma. Comprehensive review for energy efficient hierarchical routing protocols on wireless sensor networks. *Wireless Networks*, 25(3): 1159–1183, 2019.

Martin Haenggi and Daniele Puccinelli. Routing in ad hoc networks: a case for long hops. *IEEE Communications Magazine*, 43(10):93–101, 2005.

Dang Thanh Hai, Trong Le Vinh, et al. Novel fuzzy clustering scheme for 3d wireless sensor networks. *Applied Soft Computing*, 54:141–149, 2017.

Jumana Haimour and Osama Abu-Sharkh. Energy efficient sleep/wake-up techniques for iot: A survey. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, pages 478–484. IEEE, 2019.

Nguyen Thi Hanh, Huynh Thi Thanh Binh, Nguyen Van Son, and Phan Ngoc Lan. Minimal node placement for ensuring target coverage with network connectivity and fault tolerance constraints in wireless sensor networks. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2923–2930. IEEE, 2019.

Xingxing Hao and Jing Liu. A multiagent evolutionary algorithm with direct and indirect combined representation for constraint satisfaction problems. *Soft Computing*, 21(3): 781–793, 2017.

Liang He, Yu Gu, Jianping Pan, and Ting Zhu. On-demand charging in wireless sensor networks: Theories and applications. In *2013 IEEE 10th International Conference on Mobile Ad-hoc and Sensor Systems*, pages 28–36. IEEE, 2013.

Shibo He, Jiming Chen, Fachang Jiang, David KY Yau, Guoliang Xing, and Youxian Sun. Energy provisioning in wireless rechargeable sensor networks. *IEEE transactions on mobile computing*, 2012.

Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd annual Hawaii international conference on system sciences*, pages 10–pp. IEEE, 2000.

André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *arXiv preprint arXiv:1911.09539*, 2019.

Frank K Hwang and Dana S Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM transactions on networking*, 11(1):2–16, 2003.

Guiyuan Jiang, Siew-Kei Lam, Yidan Sun, Lijia Tu, and Jigang Wu. Joint charging tour planning and depot positioning for wireless sensor networks using mobile chargers. *IEEE/ACM Transactions on Networking*, 2017.

Brad Karp and Hsiang-Tsung Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, 2000.

Amar Kaswan, Abhinav Tomar, and Prasanta K Jana. An efficient scheduling scheme for mobile charger in on-demand wireless rechargeable sensor networks. *Journal of Network and Computer Applications*, 114:123–134, 2018.

Amar Kaswan, Prasanta K Jana, and Sajal K Das. A survey on mobile charging techniques in wireless rechargeable sensor networks. *IEEE Communications Surveys & Tutorials*, 24(3):1750–1779, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

R Kingsy Grace and S Manju. A comprehensive review of wireless sensor networks based air pollution monitoring systems. *Wireless Personal Communications*, 108(4): 2499–2515, 2019.

Andreas Konstantinidis and Kun Yang. Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific moea/d. *Applied Soft Computing*, 11(6):4117–4134, 2011.

Andre Kurs, Aristeidis Karalis, Robert Moffatt, John D Joannopoulos, Peter Fisher, and Marin Soljačić. Wireless power transfer via strongly coupled magnetic resonances. *science*, 2007.

Van Quan La, Phi Le Nguyen, Thanh-Hung Nguyen, Kien Nguyen, et al. Q-learning-based, optimized on-demand charging algorithm in WRSN. In *2020 IEEE 19th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE, 2020.

Sookyoung Lee, Mohamed Younis, and Meejeong Lee. Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance. *Ad Hoc Networks*, 24: 1–19, 2015.

Yu Li. An effective implementation of a direct spanning tree representation in gas. In *Workshops on Applications of Evolutionary Computation*, pages 11–19. Springer, 2001.

Wei Liang, Chaofan Ma, Meng Zheng, and Longxiang Luo. Relay node placement in wireless sensor networks: from theory to practice. *IEEE Transactions on Mobile Computing*, 20(4):1602–1613, 2019.

Chi Lin, Jingzhe Zhou, Chunyang Guo, Houbing Song, Guowei Wu, and Mohammad S Obaidat. Tsca: A temporal-spatial real-time charging scheduling algorithm for on-demand architecture in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 17(1):211–224, 2017.

Chi Lin, Yu Sun, Kai Wang, Zhunyue Chen, Bo Xu, and Guowei Wu. Double warning thresholds for preemptive charging scheduling in wireless rechargeable sensor networks. *Computer Networks*, 148:72–87, 2019.

Deyu Lin, Quan Wang, Weidong Min, Jianfeng Xu, and Zhiqiang Zhang. A survey on energy-efficient strategies in static wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 17(1):1–48, 2020.

Errol L Lloyd and Guoliang Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, 2006.

Xiao Lu, Ping Wang, Dusit Niyato, Dong In Kim, and Zhu Han. Wireless charging technologies: Fundamentals, standards, and network applications. *IEEE communications surveys & tutorials*, 18(2):1413–1452, 2015.

Zengwei Lyu, Zhenchun Wei, Jie Pan, Hua Chen, Chengkai Xia, Jianghong Han, and Lei Shi. Periodic charging planning for a mobile wce in wireless rechargeable sensor networks based on hybrid pso and ga algorithm. *Applied Soft Computing*, 2019.

Chaofan Ma, Wei Liang, Meng Zheng, and Hamid Sharif. A connectivity-aware approximation algorithm for relay node placement in wireless sensor networks. *IEEE Sensors Journal*, 16(2):515–528, 2015.

Chaofan Ma, Wei Liang, and Meng Zheng. Delay constrained relay node placement in two-tiered wireless sensor networks: A set-covering-based algorithm. *Journal of Network and Computer Applications*, 93:76–90, 2017.

Chaofan Ma, Wei Liang, Meng Zheng, and Bo Yang. Relay node placement in wireless sensor networks with respect to delay and reliability requirements. *IEEE Systems Journal*, 13(3):2570–2581, 2018a.

Yu Ma, Weifa Liang, and Wenzheng Xu. Charging utility maximization in wireless rechargeable sensor networks by charging multiple sensors simultaneously. *IEEE/ACM Transactions on Networking*, 26(4), 2018b.

Satyajayant Misra, Seung Don Hong, Guoliang Xue, and Jian Tang. Constrained relay node placement in wireless sensor networks: Formulation and approximations. *IEEE/ACM Transactions on networking*, 18(2):434–447, 2009.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2016.

Anand Nayyar, Surbhi Garg, Deepak Gupta, and Ashish Khanna. Evolutionary computation: theory and algorithms. In *Advances in swarm intelligence for optimizing problems in computer science*, pages 1–26. Chapman and Hall/CRC, 2018.

Charles C Palmer and Aaron Kershenbaum. Representing trees in genetic algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 379–384. IEEE, 1994.

Sandeep Pirbhulal, Heye Zhang, Md Eshrat E Alahi, Hemant Ghayvat, Subhas Chandra Mukhopadhyay, Yuan-Ting Zhang, and Wanqing Wu. A novel secure iot-based smart home automation system using a wireless sensor network. *Sensors*, 17(1):69, 2016.

V Prem Prakash, C Patvardhan, and Anand Srivastav. A novel hybrid multi-objective evolutionary algorithm for the bi-objective minimum diameter-cost spanning tree (bi-mdcst) problem. *Engineering Applications of Artificial Intelligence*, 87:103237, 2020.

Rahul Priyadarshi, Bharat Gupta, and Amulya Anurag. Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues. *The Journal of Supercomputing*, 76(9):7333–7373, 2020.

Heinz Prüfer. Neuer beweis eines satzes über permutationen. *Arch. Math. Phys*, 27(1918): 742–744, 1918.

Bushra Qureshi, Sammah Abdel Aziz, Xingfu Wang, Ammar Hawbani, Saeed Hamood Alsamhi, Taiyaba Qureshi, and Abdulbary Naji. A state-of-the-art survey on wireless rechargeable sensor networks: Perspectives and challenges. *Wireless Networks*, 28(7): 3019–3043, 2022.

Günther R Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, volume 1, pages 104–111. IEEE, 2000.

Günther R Raidl and Bryant A Julstrom. Edge sets: an effective evolutionary coding of spanning trees. *IEEE Transactions on evolutionary computation*, 7(3):225–239, 2003.

Jennifer S Raj, Abul Basar, et al. Qos optimization of energy efficient routing in iot wireless sensor networks. *Journal of ISMAC*, 1(01):12–23, 2019.

Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11. IEEE, 2015.

Franz Rothlauf. *Representations for genetic and evolutionary algorithms*. Springer, 2006.

Franz Rothlauf, David E Goldberg, and Armin Heinzl. Network random keys—a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1):75–97, 2002.

P Sanjeevi, S Prasanna, B Siva Kumar, G Gunasekaran, I Alagiri, and R Vijay Anand. Precision agriculture and farming using internet of things based on wireless sensor network. *Transactions on Emerging Telecommunications Technologies*, 31(12):e3978, 2020.

Jason R Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, Air Force Inst of Tech Wright-Patterson AFB OH, 1995.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Faisal Karim Shaikh and Sherali Zeadally. Energy harvesting in wireless sensor networks: A comprehensive review. *Renewable and Sustainable Energy Reviews*, 55:1041–1054, 2016.

Hemmat Sheikhi, Mohamad Hoseini, and Masoud Sabaei. k-connected relay node deployment in heterogeneous wireless sensor networks. *Wireless Personal Communications*, 120(4):3277–3292, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Nguyen Thi Tam, Dang Thanh Hai, et al. Improving lifetime and network connections of 3d wireless sensor networks based on fuzzy clustering and particle swarm optimization. *Wireless Networks*, 24(5):1477–1490, 2018.

Nguyen Thi Tam, Huynh Thi Thanh Binh, Vi Thanh Dat, Phan Ngoc Lan, et al. Towards optimal wireless sensor network lifetime in three dimensional terrains using relay placement metaheuristics. *Knowledge-Based Systems*, 206:106407, 2020.

Abhishek Tripathi, Hari Prabhat Gupta, Tanima Dutta, Rahul Mishra, KK Shukla, and Satyabrat Jit. Coverage and connectivity in wsns: A survey, research issues and challenges. *IEEE Access*, 6:26971–26992, 2018.

Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

Abhishek Verma, Virender Ranga, and Suneer Angra. Relay node placement techniques in wireless sensor networks. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1384–1389. IEEE, 2015.

Shanu Verma, Millie Pant, and Vaclav Snasel. A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems. *Ieee Access*, 9:57757–57791, 2021.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

Yin Wu and Wenbo Liu. Routing protocol based on genetic algorithm for energy harvesting-wireless sensor networks. *IET Wireless Sensor Systems*, 3(2):112–118, 2013.

Wenzheng Xu, Weifa Liang, Haibin Kan, Yinlong Xu, and Xinming Zhang. Minimizing the longest charge delay of multiple mobile chargers for wireless rechargeable sensor networks by charging multiple sensors simultaneously. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 881–890. IEEE, 2019.

Halil Yetgin, Kent Tsz Kan Cheung, Mohammed El-Hajjar, and Lajos Hanzo Hanzo. A survey of network lifetime maximization techniques in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 19(2):828–854, 2017.

Qun Zhao and Mohan Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 2008.

Jinqi Zhu, Yong Feng, Ming Liu, Guihai Chen, and Yongxin Huang. Adaptive online mobile charging for node failure avoidance in wireless rechargeable sensor networks. *Computer Communications*, 126:28–37, 2018.

Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, 103, 2001.