SAINT-PETERSBURG UNIVERSITY

Published in manuscript form

Sun Qiushi

LEARNING OPTIMAL RESOURCE ALLOCATION IN WIRELESS COMMUNICATION SYSTEMS

Scientific specialty 1.2.2 Mathematical modeling, numerical methods and software packages

DISSERTATION

Thesis for a degree candidate of
technical sciences

Scientific advisor:

Doctor of physical and mathematical sciences, professor

Petrosian Ovanes Leonovich

Saint-Petersburg

2024

Contents

Introduction

Relevance of Thesis Topic

Wireless communication entails data transfer between a minimum of two devices without the need for physical cables; instead, it relies on establishing connections through radio waves. The proliferation of computers, cell phones, and tablets has emphasized the vital role of mobile communication systems, enabling connections between mobile devices and transmitters like Access Points (APs) and Base Stations (BSs). Over the past decades, mobile communication systems have evolved from the first generation (1G) to the fourth generation (4G), with ongoing developments in fifth generation (5G) and Beyond 5G (B5G) communication networks. Unlike 4G networks, the primary goal of 5G and B5G networks is to achieve a capacity increase of up to 1,000 times, enabling speeds of up to 10 Gbps. These advanced networks also aim to minimize latency to almost imperceptible levels and establish ubiquitous connectivity, among other noteworthy features. Compared to existing 4G technology, 5G systems exhibit enhanced spectral efficiency, allowing for higher data transmission rates in a given area. Furthermore, 5G systems enhance communication reliability, supporting a larger number of simultaneous device connections while consuming less power. They also facilitate a higher number of concurrent and instantaneous device connections. The concept of the Internet of Things (IoT) refers to a platform that fosters collaborative connectivity among a wide array of devices.

Numerous studies have acknowledged the significant challenge posed to network resource management due to the exponential increase in the quantity of network devices and data generated. Several technological measures have been suggested to address this issue. These measures include strategies for efficient resource allocation, advancements in data compression algorithms, and improvements in channel coding systems. This thesis focuses on optimizing resource allocation techniques while recognizing the complexity and impracticality of resolving all approaches. In a wireless communication system, resources typically encompass bandwidth, power, frequency, and time. The

capacity of a wireless communication system is constrained by both the available resources and the resource allocation method employed by the transmitter. These factors collectively determine the amount of information that can be received by a receiver. Various resource allocation techniques offer varying system performances. The primary goal of an efficient resource allocation method is to judiciously distribute limited resources to recipients, thereby maximizing the utilization of these scarce resources for optimal system performance. Consequently, the development of a proficient resource allocation strategy is of paramount importance.

Overview of Previous Research

Resource optimization in wireless networks, such as power allocation and beamforming, is often formulated as Mixed Integer Nonlinear Programming (MINLP) problems, which are difficult to solve and are usually NP-hard problems. Various optimization and learning techniques have been proposed to obtain solutions to MINLP problems, some of which are listed below:

Model-driven approaches: Fractional Programming (FP) offers a valuable approach by suggesting quadratic transformations, simplifying ratio-based optimization. It achieves this by converting the original non-convex problem into a sequence of convex problems [2]. Another approach, Weighted Minimum Mean-Square Error (WMMSE), leverages local channel information to identify optimal points that maximize both rate and weight [3]. Nevertheless, most existing algorithms are primarily designed as partially optimal solutions. While these algorithms perform admirably in simulation experiments, their practical implementation in industrial scenarios remains challenging. Their effectiveness heavily relies on analytical and efficiently solvable mathematical models, which can be difficult to construct in real-world settings due to specific user distributions and geographic environments.

Meta-heuristic algorithms: Meta-heuristic algorithms are widely employed stochastic search techniques, offering a robust solution for solving intricate problems. They excel in situations where precise mathematical models are not readily available, as they approach optimization problems as black boxes [7]. In the domain of optimal power allocation for communication systems, various state-of-the-art methods have been developed. Adaptive Particle Swarm Optimization (PSO) is tailored for resource allocation in networked wireless sensors [8]. Additionally, an enhanced single-stage Artificial Bee Colony (ABC) algorithm has been devised to address resource allocation challenges in D2D communication networks [9]. These algorithms are favored for their relatively

straightforward implementation, self-sufficiency from detailed system information, and adaptability to derive practical solutions in real-time. However, it's worth noting that their performance can be sensitive to specific parameters, potentially necessitating adjustments for different use cases.

Machine learning (ML): ML have gained substantial traction thanks to the exponential growth in wireless devices and the rich datasets they generate. Additionally, the accessibility of high-performance computing tools, including GPUs, has accelerated the training of these algorithms. Machine learning can be broadly categorized into deep learning and reinforcement learning (RL), depending on the training process. Deep learning involves autonomous systems that utilize data to uncover patterns and make predictions [4]. This domain can be further divided into supervised and unsupervised deep learning. In supervised learning, labeled data samples are employed to learn mappings within classical optimization algorithms' input-output spaces. For instance, Deep Neural Networks (DNNs) are leveraged to approximate the input-output mapping present in conventional WMMSE algorithms [32]. On the other hand, unsupervised learning utilizes neural networks to parameterize the resource allocation function, employing the optimization objective as the loss function directly. This eliminates the need for solving specific problem instances and does not rely on pre-labeled samples. However, it's important to note that unsupervised methods may require longer training times for model parameterization compared to supervised approaches [6]. However, traditional deep learning methods do not consider the topology of the network, and the model requires a large amount of sample data to be trained to achieve satisfactory performance.

Graph Neural Networks (GNNs): GNNs have emerged as valuable tools for addressing non-Euclidean structured data in communication network problems, offering efficient use of domain knowledge and the ability to capture spatial information concealed within network topology. In [11], link scheduling techniques are achieved through graph embedding rather than relying on perfect Channel State Information (CSI), demonstrating robust performance even with limited datasets. Message Passing Graph Neural Network (MPGNN) was introduced to tackle resource allocation issues. In [12], the WMMSE algorithm enhances convergence by integrating a learning GNN module. In response to the inherent heterogeneity in modern networks, researchers have developed the Heterogeneous Graph Neural Network (HetGNN) model for solving RA challenges in heterogeneous networks. In [13], the Heterogeneous Interference Graph

Neural Network (HIGNN) was devised to learn Power Control and Beamforming policies. The communication links between transmitters and receivers are treated as distinct node types, and HetGNN was employed to learn power control policies. Nonetheless, many existing deep models overlook both node and edge features, leaving room for improvement in their performance, particularly when the size of the hidden network layer surpasses that of the node and edge features.

Reinforcement learning (RL): RL is an autonomous computational framework that develops decision-making abilities through a process of iterative experimentation and subsequent evaluation of outcomes. An agent acquires these decision-making skills through its interaction with the surrounding environment [14]. Both deep learning and reinforcement learning are autonomous learning systems. The difference is that deep learning learns patterns from a training set and applies the learning to a new set. In contrast, reinforcement learning learns to make decisions dynamically based on feedback from interaction with the environment. Note that deep learning and reinforcement learning can work together. In RL, deep neural networks can be used to learn Q-table mappings, resulting in Deep Reinforcement Learning (DRL) [53].

Summarizing the above, it is essential and significant to investigate the impact of heterogeneity, incomplete information and random factors on network formation, in particular the stable network topologies, players' interaction patterns along the dynamics, cooperative behavior as well as the equilibrium structure.

The Goal of the Thesis

The main goal of this thesis is to create a set of resource optimization algorithms that use artificial intelligence and optimization methods to manage and control resources like power and beamforming while taking into account multiple constraints in real-time decision-making situations in order to make the best use of all resources in the underlying network. Based on more realistic constraints, the proposed online algorithm allocates the available resources to the bandwidth and end-to-end delay along the routing path from the source node to the destination node. In addition, the benefits of the newly proposed algorithm will be seen in creating real-time requirements for delay-sensitive 5G applications, in addition to solving the problem of resource allocation for large-scale networks, using fewer resources, and incurring lower costs. In addition, the proposed algorithm can adapt to various Quality of Service (QoS) requirements, guaranteeing high QoS levels and providing high access for higher priority classes in congestion scenarios.

Tasks Set to Achieve the Goal

To achieve the overarching objectives of the research, the following specific goals have been formulated, each requiring corresponding tasks for resolution:

1. To construct a mathematical model of the resource allocation problem in Cartesian coordinate system for a variety of complex scenarios of communication networks (cellular networks, D2D networks, large-scale ultra-dense networks) that conforms to the control principles of the communication system, and is used to simulate the operating parameters and performance of the system in a realistic environment.

2. To improve and optimize existing meta-heuristic optimization algorithms based on stochastic search for optimizing the resource allocation problem in homogeneous cellular networks based on black-box environments.

3. To develop a traditional deep learning based algorithmic framework for optimizing the resource allocation problem for homogeneous D2D networks with structured data.

4. To develop an algorithmic framework based on graph deep learning for optimizing the resource allocation problem for heterogeneous D2D networks with unstructured data.

5. To develop a reinforcement learning based algorithmic framework for optimizing the resource allocation problem of heterogeneous D2D networks with large-scale hyper-dense nature.

Scientific Novelty

The scientific novelty of the thesis lies in the fact that the implemented research and applied analysis provide new solutions to problems related to resource allocation and interference management in wireless networks. The scientific novelty of the thesis research results can be categorized as follows:

1. The heuristic algorithm most suitable for the problem of resource allocation in communication networks is obtained after comparative tests. A new resource allocation method based on this heuristic algorithm and deep learning techniques is proposed.

2. The wireless communication network is formulated as a graph optimization problem. The new method based on graph neural network is proposed to train the model in supervised and unsupervised manner, respectively.

3. A new GNN-algorithm is proposed for graph-structured heterogeneous communication networks. The algorithm stands out as the pioneer in concurrently considering the edge features within communication interference graphs and the heterogeneous characteristics of graph elements. It uniquely harnesses both the distinctive edge features and the inherent heterogeneity to augment the learning capabilities of graph neural networks.

4. A multi-type multi-agent reinforcement learning algorithm based on the mean field theory is proposed for large-scale ultra-dense networks. This algorithm conceptualizes transceiver pairs as agents, organizing them into groups based on their connection types. It employs multi-type mean field reinforcement learning to train these agents, aiming to derive the optimal policy. Notably, this marks the inaugural application of multi-type, multi-intelligent mean-field reinforcement learning to address optimization challenges within communication networks.

Methodology and research technique

The research follow conventional research methodologies commonly used in dissertations. These methodologies encompassed various steps, including literature review, goal setting, selection of appropriate software tools and problem-solving techniques, software development, experimental testing, and evaluation of the developed solutions. Furthermore, the study involved the analysis of results obtained from these processes.

In the theoretical section of the thesis, multiple methodologies were applied. These included the analysis and synthesis of architectural solutions, the use of information systems, software, and interaction design techniques, the application of object-oriented and functional programming methods, the use of tools and methodologies for experimental software testing, the implementation of algorithmic modeling, and the application of programming techniques for wireless networks. The practical component of the research focused on the application of algorithmic models and communication system techniques.

Theoretical and Practical Significance

In practical applications, the study of wireless communication network optimization and resource allocation strategies is of great significance for improving network quality and service level. Specifically, it can help communication operators better understand user needs and provide users with more stable, secure and fast network services. At the same time, it can also reduce the cost of hardware network resources, improve the economic efficiency and competitiveness of the operator, so as to better meet the market

demand.

Theoretically, we improve the existing algorithms and also develop a new framework of optimization algorithms for scenario-specific complex communication systems. At the same time, we investigate the nature of the newly developed algorithms. The convergence of the algorithms is demonstrated through the use of mathematical proofs and complexity analysis, and the robustness of the algorithms is tested through theoretical analysis and experiments on different data sets.

Verification of Results

The main results of this paper were presented at International Conferences “Stability and control processes” (Saint Petersburg, 2021); International Conferences “International Conference on Swarm Intelligence” (Shenzhen, 2023).

Publications

Based on the results of the thesis, the following works were published: [7, 84, 85, 87]. The following items [7, 87] are published in peer-reviewed journals from the list of the Higher Attestation Commission. And [88] is accepted.

Acknowledgments

The author express his deep gratitude to Ovanes Petrosian, Doctor of Physical and Mathematical Sciences, Professor of St.Petersburg State University, for his support and guidance in the research work of this paper During the preparation of this paper. The authors are grateful to their parents for their care, understanding and encouragement in various situations.

Main Result and Contributions

Motivated by the discussions above, the ML-based resource allocation scheme is a crucial enabler for realizing 5G and B5G services. By well-training the neural network model, the performance of approximate heuristic algorithms can be obtained. At the same time, it also meets the low-latency, real-time decision-making requirements of communication networks. Therefore, this paper aims to optimize the corresponding resource allocation schemes in cellular and D2D networks with ML techniques to expand the system capacity and improve the quality of service. The main contributions of each chapter are summarized as follows:

Chapter 1 investigates issues related to power control in cellular network systems. We briefly introduce the background theory of resource allocation and interference management. The formulated nonconvex problem is treated as a black box, and heuristic algorithms are used to search for approximate optimal solutions. This chapter provides

the first comprehensive comparison of today's popular heuristic algorithms, tests their performance in network resource optimization problems, and finds the algorithm with the best performance. With data-driven machine learning algorithms, the training set determines the upper bound of the model, and the heuristic algorithms in this chapter can provide near global optimal solutions for subsequent supervised learning [7].

Chapter 2 and Chapter 3 investigates more realistic designs for beamforming and power allocation in D2D networks. We combine heuristic algorithms with deep learning to propose supervised allocation algorithms based on DNNs and GNNs, respectively. The channel states and the near-optimal allocations generated by the heuristic algorithms are used as the training set for deep learning, and the models are trained to learn the mapping of features to optimization variables in a supervised manner. We also consider the network's topology to learn the resource allocation strategy in a graph learning approach [85,88].

Chapter 4 investigates the heterogeneous D2D communication network scenario. The graph neural network models are trained through unsupervised learning. We propose supervised allocation algorithms based on GAT and EGAT, respectively. Compared to the spectral-domain based GNN in the previous chapter, the research in this chapter focuses on the more generalizable spatial-domain based GAT. also, to enhance the algorithms' performance, edge features are introduced to strengthen the learning process. The goal is to maximize the system sum rate by jointly optimizing the beamforming design and power allocation [87].

Chapter 5 investigates the hyperscale, dense heterogeneous communication network scenario, a typical heterogeneous multi-agent system. This chapter proposes a resource allocation algorithm based on Multi-intelligent Reinforcement Learning (MARL) with Mean Field Type Gaming (MFTG). By considering the interactions between each agent and different types of mean fields, the scalability problem of individual reward computation is overcome, which leads to efficient resource allocation.

Finally, Chapter coclusion summarizes the paper and further suggests several potential future research topics.

Research Results to be Defended

1. The new mathematical models of complex communication networks (heterogeneous edge information D2D networks, large-scale ultra-dense heterogeneous D2D networks) are described and developed. The problem of resource allocation in communication systems is constructed as a graph representation model of the net-

work and a pair-type multi-agent system to learn optimization strategies using neural network techniques. A new dynamic, interactive simulation environment of the communication system is built through computer programming.

2. The existing heuristic algorithms which designed to tackle resource allocation problems are enhanced. The improved algorithms are not reliant on a mathematical model of the system and can discover approximate optimal solutions through random search in black-box scenarios. Theirs performance surpasses that of optimization methods based on mathematical models, particularly in large-scale networks. Extensive simulation experiments have been conducted to validate the superior performance and stability of the enhanced algorithm.

3. A new algorithm for solving the resource allocation problem in homogeneous communication networks based on meta-heuristic algorithms and deep learning techniques is developed. The optimization policy obtained by the proposed algorithm through learning can meet the demand of dynamic resource allocation, and at the same time, it can achieve the performance of the approximate heuristic algorithm. The superiority and scalability of the algorithm are verified through simulation experiments.

4. A new algorithm solving the heterogeneous network resource allocation problem based on graph-based edge features and graph learning techniques is developed. The proposed algorithm enhances the learning ability of graph neural networks through edge features, and its performance exceeds that of traditional graph neural network algorithms. The superiority and scalability of the proposed algorithm are verified by simulation experiments.

5. The new multi-type mean-field reinforcement learning algorithms tailored for acquiring optimization strategies in extensive heterogeneous multi-agent communication auditory systems are developed. Various transceiver pairs are conceptualized as distinct sets within the mean field, and interaction strategies among different sets are explored through mean-field theory. The optimization strategies based on multi-type mean fields in heterogeneous agent systems are shown to work better than standard mean-field reinforcement learning methods using mathematical proofs. The efficacy and robustness of the proposed algorithm are further substantiated through simulation experiments.

Chapter 1

Resource allocation in Homogeneous Cellular Network with Metaheuristic Algorithms

This chapter considers the problem of power allocation in cellular networks. Improvements are made to existing meta-heuristic algorithms, and a series of state-of-the-art stochastic algorithms are compared with the benchmark algorithms. Simulation results demonstrate the superiority of the proposed algorithms over the traditional benchmark algorithms.

1.1 Background

1.1.1 Background Knowledge of Cellular Network

A cellular network is a mobile communications network that derives its name from how its base station coverage area is arranged, similar to a honeycomb. This type of network has a base station at its center that divides the communication area into several small hexagonal or honeycomb-shaped areas, with a single base station serving each honeycomb. Cellular networks are the basis of mobile communication systems [16]. Cellular networks have the following main components: Base Station (BS): BS is the core components of a cellular network, which is used to send and receive wireless signals. Each BS covers a small area known as a cell. Cell: A cell is a coverage area in a cellular network, usually in the shape of a hexagon or similar. A single BS serves each cell, which makes it easier to allocate communication resources efficiently. Mobile Devices: Mobile devices, such as cell phones, tablets, and smartwatches, connect to a cellular network to communicate. They are usually equipped with built-in antennas to communicate with the base station [1].

1.1.2 Background Knowledge of Resource Allocation

In the rapidly advancing landscape of wireless technology, mobile applications and services have gained significant traction in various aspects of life. These encompass a broad spectrum of applications, ranging from the streaming of 4K videos to drone operations and indoor positioning services. It is essential to recognize that these diverse applications often impose distinct demands on the underlying network infrastructure. For instance, the realm of autonomous vehicle operation necessitates ultra-low communication latency to ensure the safe and responsive movement of vehicles. Consequently, resource allocation plays a pivotal role in catering to the distinctive prerequisites of these applications.

In practical communication systems, the concept of resource allocation encompasses a broad and inclusive array of strategies. These strategies encompass the management of channel access, allocation of power resources, distribution of available bandwidth, user-device associations, energy management policies, and the design of optimal beamforming configurations. This multifaceted approach to resource allocation is indispensable for ensuring the efficient operation of communication systems in support of the diverse requirements posed by modern applications.

Resource allocation is a complex and critical issue in cellular networks, which requires a high degree of optimization and dynamic management. With proper resource allocation, cellular networks can provide efficient communication services, increase network capacity, and meet the growing demands of users. Research in this area continues to drive the development of communication technologies and improve network performance.

1.2 System Model of of Cellular Network

We consider a classical scenario of downlink multicell communication, a massive MISO network with M-antennas BSs and single-antenna UEs. The PA problem in the cellular network is with the setting of interfering multiple-access channels (IMAC) [17]. All BSs within the network coverage area simultaneously serve all UEs. However, since different cells use the same frequency, the UEs are still subject to inter-cell and intra-cell interference. Index the BSs as $\mathcal{N} = \{1, ..., N\}$ and UEs as $\mathcal{K} = \{1, ..., K\}$. Denote D_{nk} as the set of k-th UE's neighbour UEs in the n-th cell, denote C_n as the set of n-th cell's neighbour cells. Assume that $n \in \mathcal{N}$, $k \in \mathcal{K}$, $k' \in D_{nk}$, $n' \in C_n$, then the

received signal of the k-th UE from n-th BS in n-th cell can be formulated by

$$\begin{aligned} y_{nk} =& \underbrace{g_{n,nk}^H w_{nk} \sqrt{p_{nk}} s_{nk}}_{\text{desired signal}} + \underbrace{\sum_{k' \neq k} g_{n,nk}^H w_{nk'} \sqrt{p_{nk'}} s_{nk'}}_{\text{intra-cell interference}} \\ &+ \underbrace{\sum_{n' \neq n} \sum_{k'} g_{n',nk}^H w_{n'k'} \sqrt{p_{n'k'}} s_{n'k'}}_{\text{inter-cell interference}} + z_{nk}, \end{aligned} \tag{1.1}$$

where $g_{n,nk}$ denote the the channel response from n-th BS to k-th UE in n-th cell, p_{nk} denote the corresponding transmit power. $s_{nk} \sim \mathcal{U}(0,1)$ is the transmit signal. $z_{nk} \sim \mathcal{N}\left(0, \sigma^2\right)$ is the additive white Gaussian noise (AWGN). The coordinated beamforming (CB) vector from n-th BS to k-th UE is denoted as w_{nk}.

The literature on Coordinated Multi-Point (CoMP) CB has explored multiple schemes. In this study, we have opted to utilize the zero-forcing beamforming scheme to simplify the problem. Then the signal-to-interference-plus-noise ratio (SINR) of k-th UE can be calculated as

$$\gamma_{nk} = \frac{g_{nk,nk} p_{nk}}{\sum_{k' \neq k} g_{nk,nk'} p_{nk'} + \sum_{n' \neq n} \sum_{k} g_{nk,n'k'} p_{n'k'} + \sigma^2}\ , \tag{1.2}$$

where $g_{nk,nk} = \left| g_{n,nk}^H w_{nk} \right|^2$ denote independent channel gain of desired signal. $g_{nk,nk'} = \left| g_{n,nk}^H w_{nk'} \right|^2$ denote channel gain of intra-cell interference from neighbour UEs in n-th cell. $g_{nk,n'k'} = \left| g_{n',nk}^H w_{n'k'} \right|^2$ denote channel gain of inter-cell interference from neighbour UEs in n-th cell's adjacent cells.

The downlink rate of communication link nk can be expressed in terms of normalized bandwidth as

$$C_{nk} = \log_2 \left(1 + \gamma_{nk}\right). \tag{1.3}$$

The primary aim of this study is to identify the ideal power level that optimizes the overall network sum rate, while adhering to the limitation of a maximum power limit for each transmitter. The provided problem may be expressed as

$$\begin{aligned} &\max_{p_{nk}, w_{nk}} \sum_{n=1}^{N} \sum_{k=1}^{K} \log_2 \left(1 + \gamma_{nk}\right) \\ &\text{s.t. } 0 \leq p_{nk} \ \leq p_{\max}, \forall n \in \mathcal{N}, k \in \mathcal{K}. \end{aligned} \tag{1.4}$$

The objective function presents a challenging obstacle in the form of a nonconvex nonlinear optimization problem, which is further complicated by the presence of con-

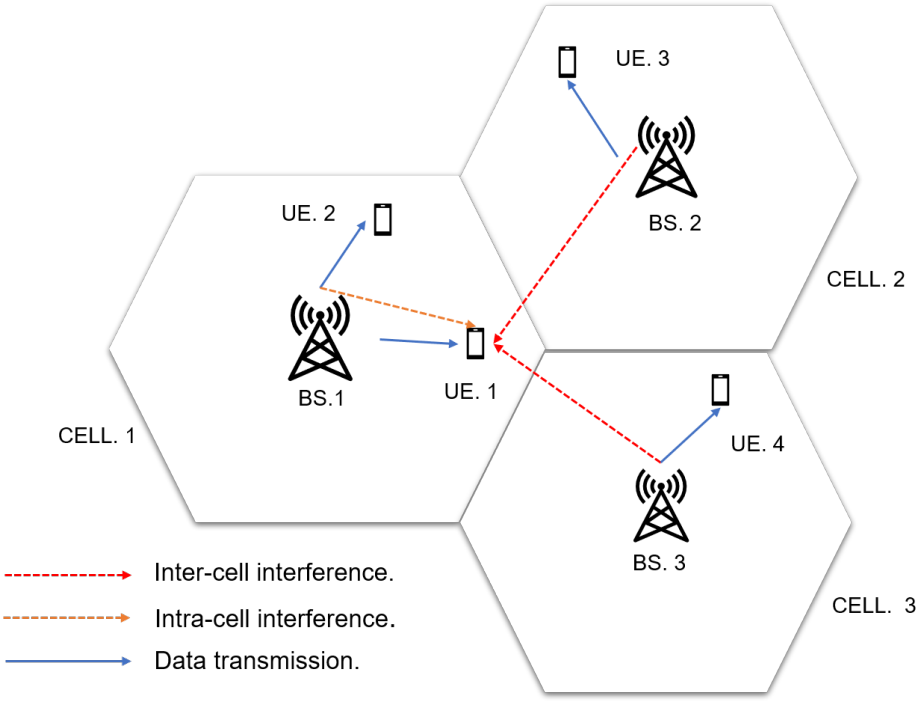

Figure 1.1: Example of wireless cellular network.

straints. As a result, finding the global optimal solution becomes a complex endeavor. Heuristic algorithms has the capability to approximate solutions that are globally optimum, but at the cost of substantial computing resources. In order to cater to the demand for real-time applications that require low-complexity solutions, we suggest using deep learning techniques to parameterize these solutions. Although attaining theoretical optimality for learnt solutions may pose challenges, actual evidence continually shows that deep learning methods often produce extremely satisfying performance results.

For model-driven approaches, it is generally hard to evaluate the performance gap from the optimal solution, and practical implementation is limited because of the imperfect mathematical model. Furthermore, it is hard to adapt the model-based method to heterogeneous cellular networks because of the imperfect mathematical model in real communication scenarios. Thus, model-free metaheuristic algorithms are discussed in the following section.

1.3 Metaheuristic Algorithms

To find an efficient approach for this optimization problem, we consider various competitions in global optimization to select the best algorithms from feasible techniques. Moreover, the competitive process generates novel ideas that can be developed into practical solutions. We consider the Special Session and Competition on Large-Scale

Global Optimization in the past ten years and concentrate on different mathematical optimization methods with one objective. We select nine metaheuristics algorithms from the winning rank, such as Artificial Bee Colony (ABC), Self-adaptive Differential Evolution (jDE and iDE), Particle Swarm Optimization Generational (GPSO), Extended Ant Colony Optimization (EACO), Differential Evolution (DE), Particle Swarm Optimization (PSO), Simulated Annealing (SA), Monotonic basin hopping (MBH), Covariance matrix adaptation evolution strategy (CMA-ES). We consider nine metaheuristic algorithms from four primary types. A brief description of these algorithms is as follows:

1.3.1 Swarm Intelligence Algorithms

Artificial Bee Colony

The Artificial Bee Colony (ABC) algorithm is a stochastic search technique based on the intelligent foraging behavior of honey bee swarms. In this algorithm, each candidate solution indicates the location of the food source in the search space, and the quality of the food source is employed as a fitness evaluator.

The model involves three essential elements: employed bees, onlookers, and food sources. The amount of employee bees is equal to the food sources. Employed bees depart from the hive to search for a food source and collect information about the quality of the other food sources in the neighborhoods of discovered location. Once back in the hive, they transmit information about the explored food source to the onlookers. Onlookers evaluate a new location from the information provided by the employed bees according to the selection probability of quality and prefer the food source with high fitness value. Onlooker becomes an employed bee when it selects a new food source to explore. The employed bee switches to the scout bee and randomly searches for new food resources in the search space when its explored food source is abandoned. This process is repeated until the optimal food source is found.

Advantages of ABC: it requires few parameters, performs robustly, converges fast, and is highly flexible. Disadvantages of ABC: it may converge prematurely in the phase of its search, and the classification accuracy of the best value it obtains may not meet the requirement.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a swarm intelligence technique. The initial idea of PSO is inspired by the population behavior of bird flocking and fish schooling. PSO and evolutional strategic techniques have many standard features [18]. This algorithm simulates the behavior of members' information interaction and collaboration. The difference with the genetic algorithm is that PSO does not require evolution operators like crossover and mutation. In the model, there is a population of candidate solutions called particles. These particles move around in the search space over their position and velocity. Each particle's movement is guided toward the best-explored positions in the search space, updated as other particles find better positions. This is expected to move the swarm toward the best solutions. The PSO process can be formulated as:

$$v_{id}^{t+1} = v_{id}^{t} + c_1 * rand\,(0,1) * \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 * rand\,(0,1) * \left(p_{gd}^{t} - x_{id}^{t}\right),$$

and

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t},$$

where x_{id}^{t} and x_{id}^{t} represent the position and velocity of each particle, The parameter d is the population size, i is the index of the each particle and t is the number of iterations. c_1 and c_2 are learning factors. p_i represents value explored by ith particle, p_g represents value explored by neighbours of the ith particle. PSO can be implemented as Algorithm 1:

Algorithm 1 Particle Swarm Optimisation

Require: Generate initial individual
Ensure: The best vector
  while Termination condition not met do **do**
    for Each particle x with position p_i **do**
      Calculate fitness value
      if fitness value is greater than the current best value p_{best} **then**
        Set current best value as p_{best}
      end if
    end for
    Select the particle with the overall best fitness value and set it as g_{best}
    for Each particle **do**
      Calculate particle velocity
      Update position of particle
    end for
  end while

Advantages of PSO: it has a simple calculation without overlapping and mutation. Disadvantages of PSO: it may fall into local optimum in high-dimensional space and has a low convergence rate in the iteration.

Generational Particle Swarm Optimization

Generational Particle Swarm Optimization (GPSO) is a variant of the standard PSO algorithm [19]. In the PSO algorithm, velocity is one of the most significant parameters; if the particles' velocity in the swarm is updated effectively, no search effort will be wasted by searching in the wrong directions. The procedure of PSO is to move the particle to search for the positions of optimal solutions. The velocity at which the particles change positions is usually adjusted by multiplying the velocity by a factor. Unlike the standard algorithm, the velocity is first calculated for all particles; then the position is updated.

GPSO can handle stochastic optimization problems according to iterative random seed schema. But it is not suitable for multi-objective problems.

Extended Ant Colony Optimization

Ant Colony Optimization (ACO) is a classical bio-inspired technique which based foraging behavior of natural ants [20]. The ACO algorithm simulates the process a colony of ants seeking for the shortest path from nest to the food source. In the model, a group of simulation agents imitate the foraging behavior of natural ants to search for the minimum value of function. Each agent depart from the nest in search of food source and arrive at the nest as the end of the trial. Each agent leaves a marker which called the pheromone on the path they take in search of food source. The pheromone concentration on each path is used to evaluate the distance of the path and the quality of the food source. The information implied by the pheromone on the path plays an important role in the subsequent agent's selection to the path. The higher the fitness value of the path evaluation, the higher the probability that the path be accepted. Extended ACO improve the original algorithm by using the multi-kernel gaussian distribution which based on three parameters which are computed depending on the quality of each previous solution. The objective function value are ranked through an oracle penalty method.

Advantages of EACO: its parallel process can search solutions independently and simultaneously. Disadvantages of EACO: its probability distribution iteration changes and the convergence time is not stable.

1.3.2 Differential Evolution algorithms

Differential Evolution

The Differential Evolution (DE) algorithm is one of the most popular techniques for continuous optimization problems. DE is based on the evolution strategy but not inspired by the natural paradigm like common ones [21]. It is proposed to search for the minimum value of non-differentiable and nonlinear continuous functions. Classical DE has two significant features to be adjusted: the learning strategy and the control parameters. The learning strategy comprises the primary type of operators in genetic algorithms, such as mutation, crossover, and selection. A basic variant of the DE algorithm works by having a population of candidate solutions. These agents are moved around in the search space to combine the positions of existing agents from the population. If the value of an agent's new position is improved, it is accepted and forms part of the population. And it is excepted but not guaranteed that a global optimal solution will eventually be found.

In the mutation, a mutant vector is generated as follows:

$$v_{i,G+1} = x_{r,G} + F\left(x_{r,G} - x_{r,G}\right)$$

where F represent the scaling factor, G is the number of iterations. $x_{r,1}$, $x_{r,2}$, and $x_{r,3}$ are random searched vectors in current iteration. In the crossover, a trail vector is produced by combining the parent vector with a mutated vector.

$$u_{i,G+1} = \begin{cases} v_{i,G+1} & if\ rand_j \leqslant Cr \\ x_{r,G} & if\ rand_j > Cr \end{cases}$$

where Cr represent the crossover rate. j is random number in the resulting array. v_i is current best value, x_i is best searched value. DE can be implemented as Algorithm 2:

Advantages of DE: it can handle optimization problems with high computational complexity. Disadvantages of DE: it requires parameter tuning and its convergence is not stable.

Variants of Differential Evolution

These are two different variants of the DE algorithm based on the mechanism of self-adaptation. The learning strategies and control parameters involved in the standard DE algorithm highly rely on the specific optimization problem. This process may cost amount of time to select the strategy and adjust the parameters to make the model have

Algorithm 2 Differential Evolution

Require: Generate initial population;
Ensure: The best vector;
  while Termination condition not met do **do**
    for Each solution x_i in population **do**
      Generate new solution s_i;
      if fitness((s_i)) >= fitness((x_i)) **then**
        Retain s_i in population;
      else
        Retain x_i in population;
      end if
    end for
    Evaluate fitness of the new population
    Update the best solution
  end while
  Return best solution

a good performance. Many different proposals have been made to self-adapt both the CR and the F parameters of the original differential evolution algorithm. There are many different proposals that have been proved to adapt the CR and F parameters. The first variant (jDE) does not use the DE operators to update parameters F and CR, the procedure is more like parameter control rather than self-adaptation [22]. The second variant (iDE) uses a variation of the selected DE operator to update CR and F parameters for each individual [23].

1.3.3 Random Search Algorithms

Simulated Annealing

Simulated Annealing is a stochastic global search optimization technique to search. This algorithm emulates the statistical annealing procedure of the crystals growing to reach the global optimal internal energy configuration [24]. The annealing process works by first exciting the atoms in the material at a high temperature. This step can push atoms to heat up and accelerate their motion. And next step is to slowly cool down the temperature to reduce their excitability, making atoms convert into a more stable configuration. The essential component to implementing this simulated annealing process is to initialize a random solution in the neighborhood of the current optimal solution and evaluate the objective functions. Once the fitness value of the cost function is smaller than its current best value, the solution is accepted, and the new best

fitness value is updated. Once the fitness value is higher than the current best value, the point is accepted or rejected with probability. A parameter temperature is introduced to calculate the probability. In the cooling schedule, the temperature is reduced with the acceptance probability converging to zero. And the whole annealing process is terminated after a large number of trials. This strategy avoids being trapped in the local optimal solution.

Advantages of SA: it can handle the problem with arbitrary systems and cost functions. Disadvantages of SA: it requires parameter tuning and is possible to be trapped into local minima.

Monotonic Basin Hopping

Monotonic Basin Hopping (MBH) is a stochastic global optimization technique. This algorithm is a two-phase approach that combines the global stepping algorithm with the local minimization procedure at each iteration [25]. The algorithm model uses random perturbations to jump basins and a local search algorithm to optimize each basin. The model iterates as follows: The first phase uses random perturbation to jump basins of coordinates. The second phase uses a local optimization procedure to evaluate the new coordinates and to decide accepted or reject the coordinates based on the minimized function value. This algorithms original idea is to map the objective function into searching the local minima from the initial point. This mechanism can significantly improve the efficiency of problem-solving. Main idea of MBH is mapping the objective function $f\left(x_0\right)$ into the local minima found starting from x_0, MBH can be implemented as Algorithm 3:

Algorithm 3 Monotonic basin hopping

Require: $x_0 \leftarrow$ Generate initial solution
Ensure: optimal x, $f\left(x\right)$
  $x_0 \leftarrow$ generate initial solution $g_0 = 0$ and $d_0 = 0$
  $x_0 \leftarrow$ minimize $\left(f, x_0\right)$
  repeat
    $y \leftarrow$ perturb(x)
    $y \leftarrow$ minimize $\left(f, y\right)$
    $x \leftarrow$ acceptance $\left(x, z\right)$
  until termination condition met

In this chapter, we combine this concept's original generalization, resulting in a meta-algorithm that operates on any population using a suitable algorithm. The actual method

is recovered when a population containing a single individual is used and coupled with a local optimizer.

1.3.4 Evolution Strategy Algorithms

Covariance matrix adaptation evolution strategy (CMA-ES) is a stochastic technique for involuting non-linear non-convex continuous black-box optimization problems [26]. It is based on the idea of self-adaptation in evolution strategies. The mechanism of this algorithm is to construct parametric distribution on the searching space in which feature functions are defined in advance. A population of solution candidates is selected from this parametric searching distribution. Then, these candidates are evaluated by a black-box function. Given the function values at the sampled points, updating and storing the covariance matrix dominates the time and space complexity in each iteration of the algorithm. The covariance matrix where time and space complexity dominates is updated and stored at each iteration of the algorithm.

Advantages of CMA-ES: it is suitable for small-scale non- separable optimization problems Disadvantages of CMA-ES: high complexity and premature stagnation.

1.4 Simulations

1.4.1 Environment Setting

Algorithm deployment has a high requirement for low computational complexity and it is considered here. The configuration of the simulation platform is expressed as: CPU Intel i7 10750H and RAM 16 GB. A series of simulation experiments are executed to compare the performance of these meta-heuristic algorithms and to find the best algorithm for the power allocation issue.

Table 1.1: Simulation parameters of cellular network.

Notation	Simulation Parameter	Value
N	Number of BS	4, 9, 16
M	Average users per cell	2, 4, 8
K	Total Number of user	MN
f_d	The Doppler frequency	10 HZ
P_{min}	Minimum allocated power	5 dBm
P_{max}	Maximum allocated power	38 dBm
R_{min}	Inner space distance	0.01 km
R_{max}	Half cell-to-cell distance	1 km
T	Time period	20 ms
σ^2	Noise power spectral density	$-$ 114 dBm/Hz
z	Shadowing	8 dB standard deviation

We consider wireless cellular networks of different scales, with cell populations of 2×2, 3×3, and 4×4. In each cell, users distributed randomly and uniformly in range $r \in [R_{\text{min}}, R_{\text{max}}]$. The small-scale fading follows Rayleigh distribution, and the Jakes model is adopted with f_d. The large-scale fading is formula as $\beta = -120.9 - 37.6 \log_{10}(d) + 10 \log_{10}(z)$ dB according to the Long-Term Evolution (LTE) standard, where z is shadow effect element, and d is the transmitter-to-receiver distance (km). Table 2.1 collects the primary parameters of the network. The maximum number of iterations is determined as 1000 based on the simulation results. In general, Metaheuristic algorithms use randomized search techniques, in which optimization performance highly relies on the initial value and fine parameter tuning. Hence, reproducible optimization results obtained under the same conditions cannot be guaranteed. Therefore, we conducted 20 repeated trials and performed a statistical analysis of the results to compare the performance of the proposed algorithms. The performance of the compared methods is evaluated by averaging 20 trial runs.

1.4.2 Numerical Results

In this section, we present the simulation results to indicate the performance of the metaheuristic algorithms. We use a simple generation-evaluation method for the metaheuristic algorithm for tuning the parameters. A set of a priori candidate configurations is generated. Then, each of these configurations is evaluated to find its optimal configuration. Furthermore, we explore the computation capability of the above algorithms

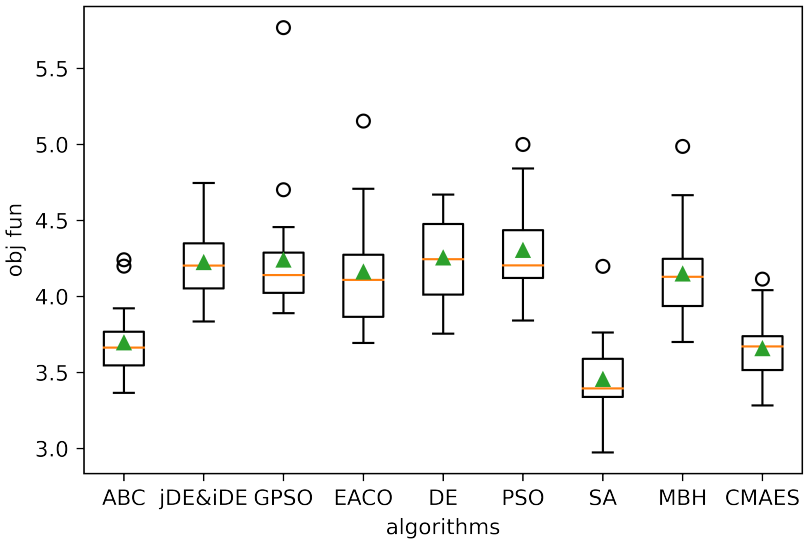

Figure 1.2: Average rate during fitness evaluation (N=4).

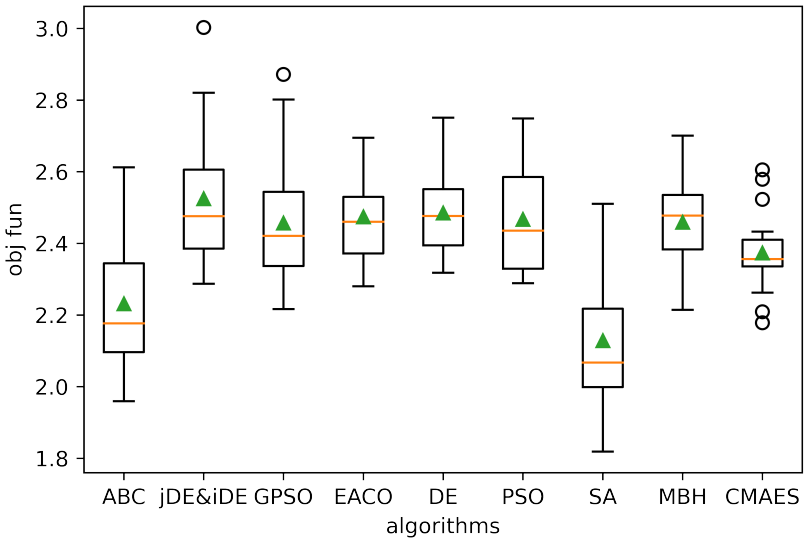

Figure 1.3: Average rate during fitness evaluation (N=9).

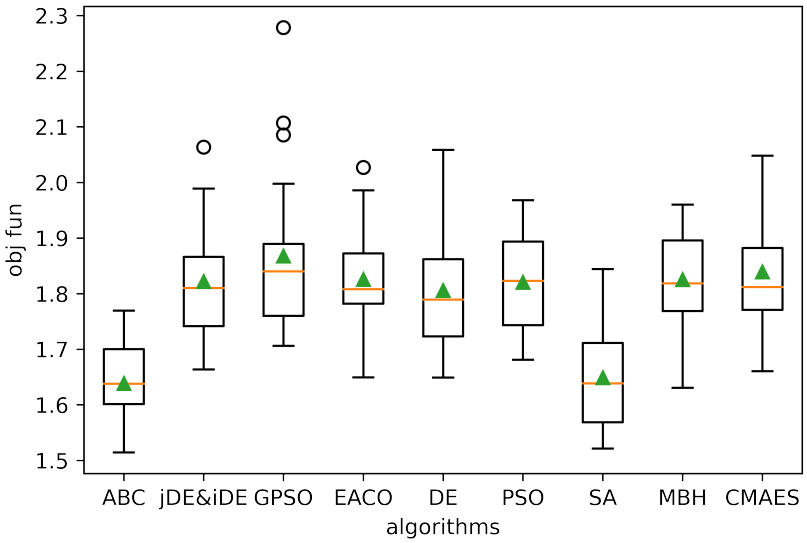

Figure 1.4: Average rate during fitness evaluation (N=16).

on different network scales. And the length of the search process is 50000 evaluations. Meanwhile, three benchmark algorithms which are FP, WMMSE, and random strategy (RAND) stated before are tested as comparisons.

Figure 1.2, 1.3 and 1.4 indicate the searching process for metaheuristic algorithms with different N values. The network's average sum rate is expressed as the value of the objective function based on the number of fitness evaluations. According to the rate of rising fitness value, we intercept two intervals from the searching range, which are [8000, 15000] and [37000, 43000]. We can observe that the Differential Evolution algorithms always have good performance when the rapidly rising, and the Swarm Intelligence Algorithms always have good performance when the slow rising. The results statistically indicate that the performances of the proposed algorithms are similar after fixed generations.

Table 1.2 shows Obtained solution of the numerical experiments.We focus on the average performance of the above algorithms for 20 trials. Based on the previous results, Swarm Intelligence Algorithms: PSO and GPSO perform the best when $N = 4$ and $N = 16$ respectively. Differential Evolution algorithms: jDE&iDE perform the best when $N = 9$.

Figure 1.5, 1.6 and 1.7 shows the corresponding distribution of the best fitness for metaheuristic algorithms. CMA-ES is the most robust technique based on the average most minor standard deviation of best values.

We also obtain a numerical example result of experiment trials with different user

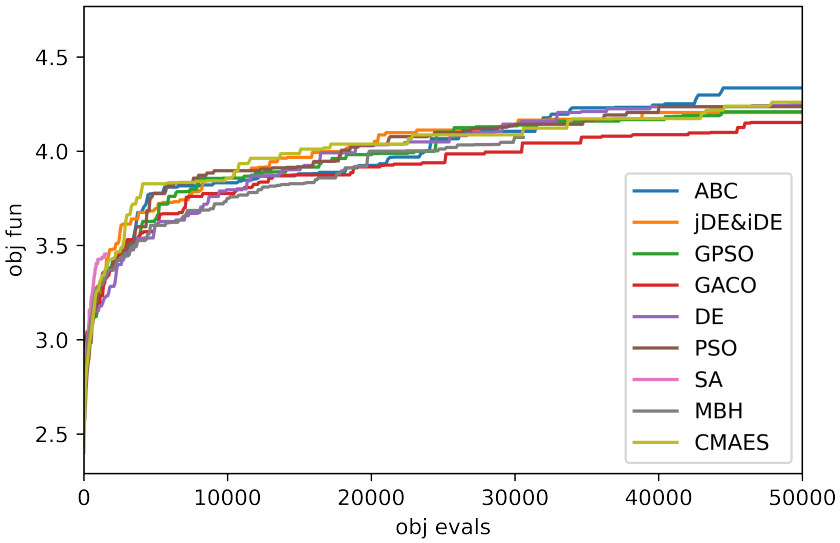

Figure 1.5: Distribution of best fitness over 20 trials. (N=4).

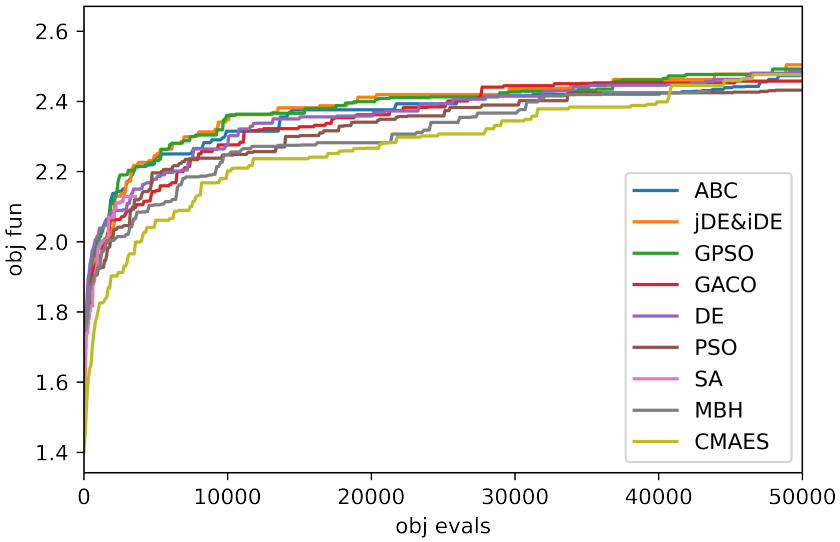

Figure 1.6: Distribution of best fitness over 20 trials. (N=9).

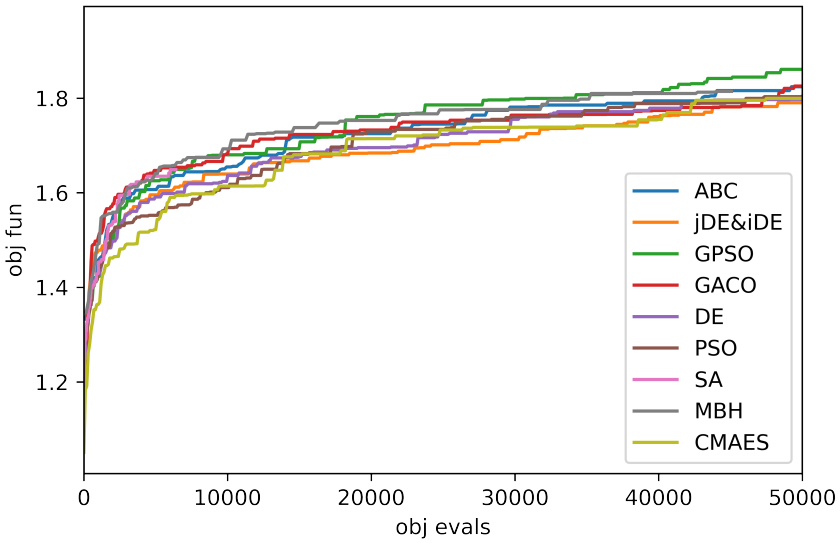

Figure 1.7: Average rate during fitness evaluation (N=16).

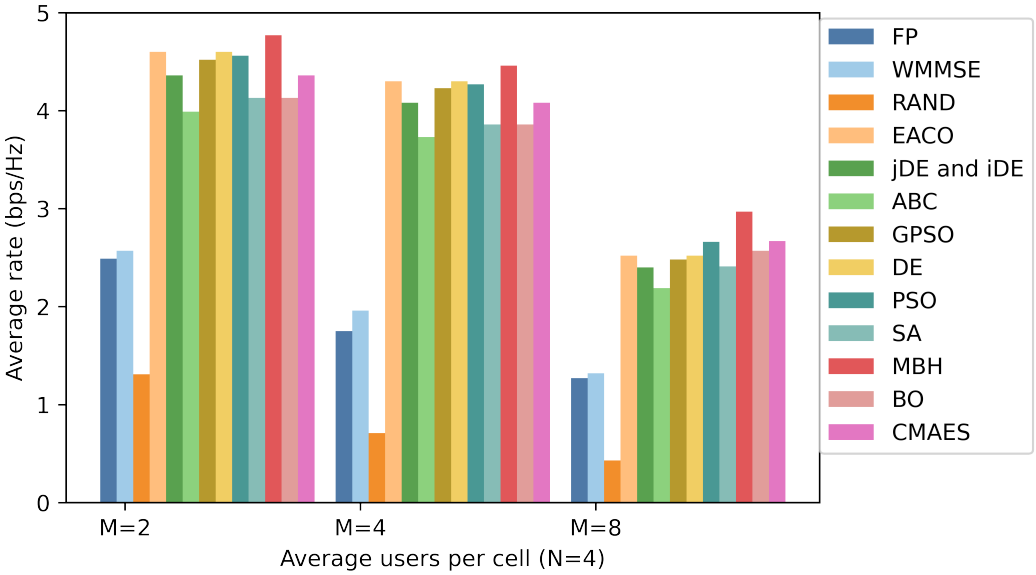

Figure 1.8: The average rate versus user number per cell.

Table 1.2: Obtained solution (bps/Hz) of the numerical experiments.

Algorithms	N=4			N=9			N=16		
	max	mean	std	max	mean	std	max	mean	std
ABC	4.241	3.694	0.224	2.612	2.231	0.197	1.769	1.638	0.067
jDE&iDE	4.747	4.221	0.222	**3.002**	**2.524**	0.181	2.063	1.821	0.093
GPSO	4.768	4.238	0.402	2.871	2.456	0.169	**2.278**	**1.867**	0.143
EACO	5.153	4.160	0.345	2.695	2.474	0.118	2.026	1.825	0.083
DE	4.670	4.252	0.275	2.750	2.484	0.122	2.058	1.805	0.100
PSO	**4.999**	**4.303**	0.313	2.748	2.467	0.151	1.968	1.820	0.089
SA	4.198	3.454	0.268	2.510	2.128	0.177	1.844	1.648	0.090
MBH	4.987	4.147	0.292	2.700	2.458	0.131	1.960	1.825	**0.083**
CMAES	4.114	3.655	**0.216**	2.605	2.373	**0.103**	2.048	1.839	0.086

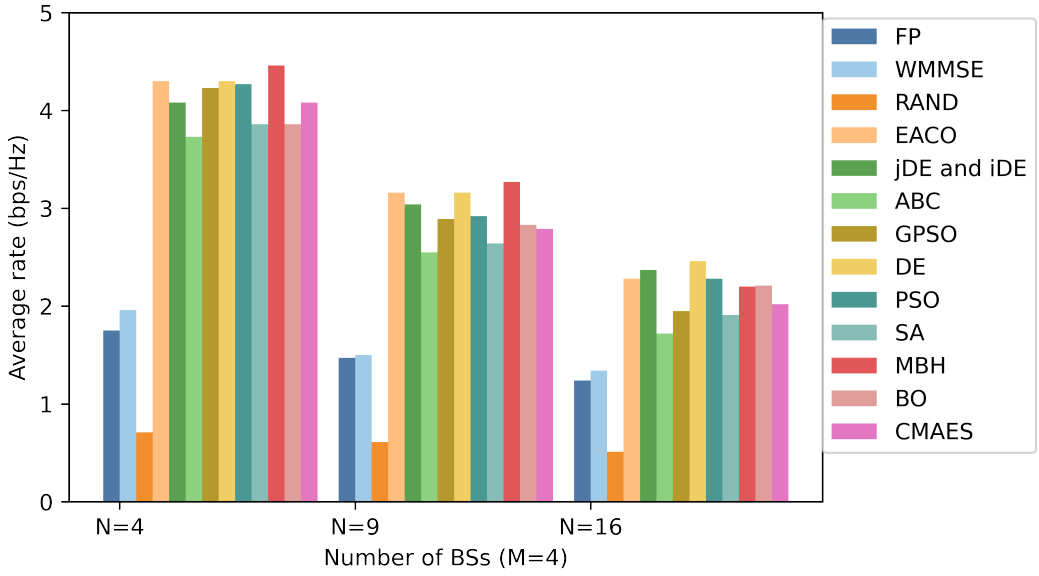

Figure 1.9: The average rate versus number of cells.

densities and scales of the network. Compared with the values of averaged sum-rate in Figure 1.8 and Figure 1.9, the performance of metaheuristic algorithms is not stable, especially depending on the specific solution scale effects. Additionally, the best fitness value of the metaheuristic algorithms decreases significantly with the increase of the solution's computation scale compared to the result of the conventional algorithm. In large-scale scenarios, this type of approach costs much more than the other algorithms over time, which means that there is still potential to improve the performance of metaheuristic algorithms.

1.5 Conclusion to Chapter 1

The optimal power allocation problem in the cellular network with IMAC has been investigated, and the model-free metaheuristic approaches have been implemented to handle this problem. To be consistent with the optimization objectives of the PA problem, the network's sum-rate SINR is used as the objective function. Then a range of metaheuristic algorithms are proposed, and these algorithms work as a black-box solver to search for the optimal power allocation under constraints with specific CSI.

The Simulation results show that the proposed meta-heuristic algorithms outperform the conventional benchmark algorithms in different scenarios. We can observe that metaheuristic algorithms have good generalization abilities with simulated communication networks. The experiment results statistically demonstrate that it is hard to determine the overall winner algorithms. The metaheuristic methods perform well generally, and the actual performance gap is related to the solution scales. Covariance matrix adaptation evolution strategy (CMA-ES) is the most robust technique. Differential Evolution algorithms (DE, jDE&iDE) and Swarm Intelligence Algorithms (GPSO, PSO) excel in general scenarios.

Chapter 2

Resource allocation in Homogeneous D2D Network with Deep Learning in Supervised Manner

In this chapter, we combines heuristic algorithms with deep neural networks to propose a resource allocation scheme called PSO-DNN. The focus is on end-to-end learning, where the input-output mapping of the problem is learned directly by considering the optimization algorithm as a black box. Simulation results show that the trained DNN-based model can provide solutions that approach the performance of heuristic algorithms and meet the requirements of real-time resource allocation.

2.1 Background

2.1.1 Background Knowledge of Device-to-Device Networks

Device-to-Device (D2D) networks are wireless communication systems where end devices establish direct connections, eliminating the need for relaying through conventional base stations or infrastructure. In this context, terminal devices establish direct links with one another, reducing communication delays and enhancing operational efficiency [27].

To optimize communication effectiveness, resources such as spectrum and bandwidth are shared among users. D2D communication has the potential to boost network capacity, reduce the burden on base stations, and accommodate more devices and users. This direct communication approach offers increased flexibility and efficiency within wireless communication systems.

Beamforming serves as a foundational technique in wireless communications for the

efficient allocation of resources and the management of interference, especially in scenarios involving multiple antennas. In the context of D2D communications, the central goal of beamforming design is to maximize the overall data transmission rate by optimizing the beamforming strategy for each D2D pair while operating within the constraints of maximum transmit power. Addressing this challenge involves tackling a continuous optimization problem.

2.1.2 Background Knowledge of Machine Learning

While heuristic algorithms offer approximate optimal solutions, their computation times still fall short of meeting the increasingly stringent time requirements of 5G and future networks. To address this challenge, machine learning techniques have gained prominence in solving optimal control problems, effectively accelerating the convergence of traditional algorithms toward accurate solutions. In this chapter, heuristic algorithms are treated as black boxes, enabling the direct learning of input-output mappings through an end-to-end learning approach [56]. End-to-end learning streamlines the process by employing a single model to capture direct mappings, eliminating the need for domain expertise and making implementation relatively straightforward.

Machine learning-based methods have proven successful in expediting resource allocation in wireless networks. These endeavors demonstrate that machine learning not only significantly reduces time complexity but also achieves high-performance outcomes closely aligned with traditional algorithms. This performance enhancement is a crucial feature, particularly relevant for the demands of 5G and other wireless communication domains.

Machine learning encompasses two fundamental learning methodologies: supervised learning and unsupervised learning. The primary distinction between these two methodologies is the need for labeled datasets for model training.

Supervised learning necessitates a substantial number of labeled datasets to facilitate the training procedure, hence demanding the expertise of individuals to provide appropriate examples [29]. The application of supervised learning has proven effective in addressing channel assignment difficulties. A neural network can be used to learn how to find a link between the features of a D2D pair (such as their geometric distances or channel state information) and a binary decision in the context of the D2D link scheduling problem. The parameters of the neural network are typically updated using the labeled outputs and targets. The cross-entropy loss function is employed to

calculate discrepancies between the measurements.

Unsupervised learning refers to a learning paradigm that does not necessitate using labeled data during the training process [30]. Unsupervised training can address complex continuous optimization problems, including power allocation and beamforming design. An unsupervised learning method can be employed to discover the optimal mapping of channel state information to beamforming in the context of D2D beamforming design. Typically, the negative sum rate is used as a loss function to determine the appropriate timing for updating the neural network. In this scenario, the use of labeled training samples is optional.

Since this chapter primarily focuses on machine learning-enhanced resource allocation, it begins with a brief overview of machine learning fundamentals. This introduction covers distinctions between supervised and unsupervised learning, classical machine learning techniques, and key metrics for assessing learning performance.

2.2 System Model of D2D Networks

We investigated a single-cell D2D network like Figure 2.1 in which several transceiver pairs compete for a fixed amount of bandwidth B. We assume that a single data stream may be sent and received simultaneously across each connection. This throughput optimization problem aims to design a beamformer transmitter for each data stream on each live connection [31]. Consider there are $K = \{1, ..., k\}$ communication links; in other words, k transmit antennas serve k single-antenna user equipment (UEs). The channel response from the link transmitter j to the receiver i is h_{ij}. i and j is index of receiver and transmitter. Let x_i be the beamforming vector for the i connection. Accordingly, the signal received at receiver i is the superposition of signals from numerous transmitters, as described by

$$y_i = h_{ii}^H x_i + \sum_{j \neq i} h_{ij}^H x_j + n_i, \tag{2.1}$$

where $n_i \sim \mathcal{N}\left(0, \sigma_i^2\right)$ denotes the additive white Gaussian noise. The achievable sum rate of link i can be expressed as the function:

$$R_i\left(X\right) = W \log \left(1 + \frac{\left\| h_{ii}^H x_i \right\|_2^2}{\sum_{j \neq i} \left\| h_{ij}^H x_j \right\|_2^2 + \sigma_i^2} \right), \tag{2.2}$$

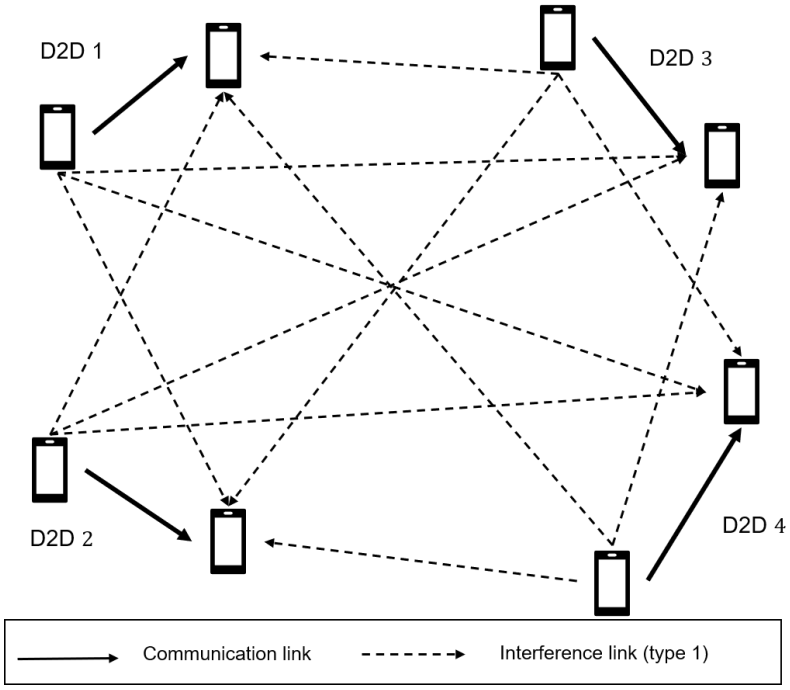

Figure 2.1: Example of D2D network with 2 transceiver pairs.

where X is a set of all beamforming vectors, and $X = (x_1, ..., x_k)$. When a single antenna is utilized at transmitters, the beamforming design simplifies to a problem of power distribution.

Typically, the aggregate performance of the communication system is determined by a utility function of the possible connection rates. Weighted sum rate is the utility function used here. Given each transmitter's power restriction, the optimization issue is formulated as

$$\begin{aligned} &\max_{X} \sum_{i} w_i R_i(X) \\ &s.t. \, \|x_i\|_2^2 \leq P_{\text{max}} \, , \forall i, \end{aligned} \tag{2.3}$$

where w_i denotes the weight of link i, and P_{max} indicate the transmit power constraint of each communication link. When all connection weights are set to 1, the problem can be considered a sum rate maximization problem.

2.3 Architecture of Learning Based Method

In this section, we build an efficient DNN-based framework for solving resource allocation issues in MIMO networks. In order to achieve a near-global optimal sum rate in real-time, we presented a two-step DNN-based power allocation technique. In the initial phase, we employ a heuristic random search approach to identify the optimal

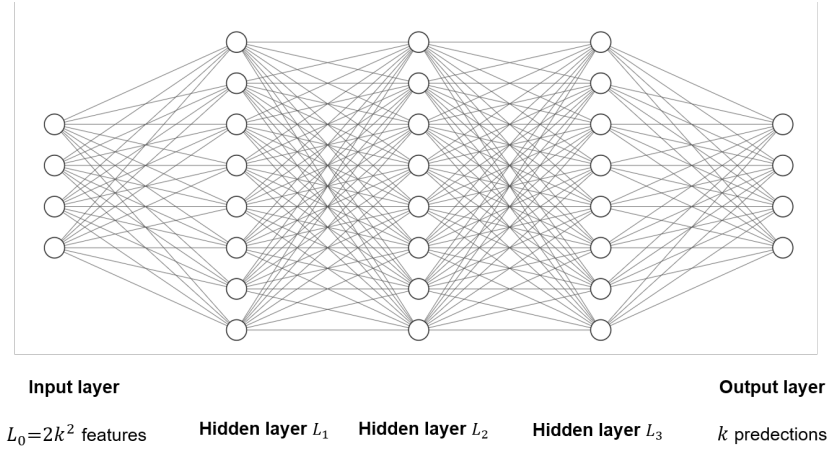

Figure 2.2: Performance versus pairwise distances.

power allocation that optimizes the system sum rate for each static channel state. In the second phase, we predict the allocated powers in real-time online applications using a well-trained DNN model. The following describes the introduction of the PSO algorithm and the design of the DNN framework for optimal beamforming:

2.3.1 Deep Neural Networks

We develop a fully-connected deep neural network (DNN) architecture that predicts the optimal resource allocation for K downlink UEs [13]. The objective is to discover a policy $P(\cdot)$ that simulates the mapping of PSO which denoted by F to estimate the optimal allocated resource $\hat{P} \triangleq \{\hat{p}_k\}$. We select a DNN parameterization of the policy $P(\cdot)$ with learnable parameters, and beamforming vectors are estimated as $\hat{P} = P_\theta(F)$.

In the first stage of the process, offline supervised learning, the computationally demanding PSO algorithm determines the best-allocated strategy and employs it as the output label. The interference relations characterized as a set of channel coefficients $\{h_{ij}\}$, the communication relation characterized as a set of channel coefficients $\{h_{ii}\}$. Note that h is a complex function consisting of a real part and a complex part. We take these two parts as features and input them into the neural network. Z_0 represents the input layer's feature vector. R is set of real number. The input feature of DNN can be formulaed as:

$$Z_0 = \left[h_{ii}^{complex}, ..., h_{ii}^{real}, ..., h_{ij}^{complex}, ..., h_{ij}^{real}, ...\right], \tag{2.4}$$

where $L_0 = 2k^2$ is the dimension of input feature. Scaling and vectorization of input features are applied at the inputs of the proposed DNN. We use the largest absolute proportion to the optimal distributed strategy, which is similar to the input features.

Similar to the the input features, we utilize the maximum absolute scaling technique for the optimal beamforming vectors in the following manner:

$$\bar{x}_k = \frac{x_k^{optimal}}{\max\left(x_1^{optimal}, ..., x_k^{optimal}\right)} \in [0, 1]\,. \tag{2.5}$$

To handle the non-linear computations, we employ the rectified linear unit (ReLU) as activation function at the hidden layers. There are L_i neurons at the i-th hidden layer, where $i = 1, 2, 3$. Z_0 is the input factor, the output of the ith hidden layer is determined as $Z_i = f_r\left(w_{i-1}Z_{i-1} + b_{i-1}\right) \in R^{L_i}$, where $f_r\left(Z\right) = \max\left(0, Z\right)$ are the weight matrix and bias vector, respectively. To match the output layer predictions between 0 and 1 as stated by the output labels, the sigmoid function $f_\sigma\left(Z\right) = \frac{1}{1+e^{-Z}}$ is used at the output layer [83]. The dimension of output facter is k. Thus, the predicted resource allocation for K downlink UEs using the DNN framework are expressed as follows:

$$[\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_k] = f_\sigma\left(W_3 Z_3 + b_3\right), \tag{2.6}$$

where W_i is the weight matrices and b_i is bias vectors which are adjusted to reduce the loss and more accurately forecast the optimal power allocation values. We evaluate the loss functions based on the predicted and ideal power values: mean square error (MSE). The formula for the MSE loss function is:

$$L_{MSE} = \frac{1}{K}\sum_{k=1}^{K}\left(\bar{x}_k - \hat{x}_k\right)^2. \tag{2.7}$$

Back-propagation is a process in which the gradient of the loss function is transmitted from the output layer to the input layer. As a result, the weight matrix W_i and the bias vector b_i are updated in order to decrease the loss, which allows for more effective learning of samples and more accurate prediction of the optimal resource allocation strategy.

2.3.2 Dataset Generation and Model Training

We create a dataset for the offline supervised learning procedure with the number of samples $S = 5 \times 10^5$. The channel gains, and UE locations with respect to the BS are randomly distributed in the area to produce the channel vector for each UE. The PSO algorithm is used to determine the associated optimally assigned powers, which are calculated and stored in the dataset.

The complete available dataset is split into 80% training and 20% validation sets during the offline learning process. After the supervised learning, a brand-new test dataset evaluates the online power allocation in time-varying scenes. The proposed algorithmic technique is implemented using open-source deep learning framework in PyTorch.

2.4 Simulation

2.4.1 Environment Setting

In order to assess the performance of the DNN-based algorithm, we primarily compare it to the WMMSE-based approach, which serves as a common benchmark in the literature for sum rate maximization problems. We also consider the following benchmarks for comparison. Please note that all results related to the test performance of WMMSE are averaged across 100 independent trials. The system settings and DNN hyperparameters is shown in Table 2.1. And the benchmarks are as follows:

- WMMSE [3]:An approach based on optimization that converts the problem of weighted mean square error reduction from the sum rate maximization difficulties in interfering broadcast channels.
- WMMSE-NN [32]: A 3-layer supervised DNN that learns the mapping of classical WMMSE.
- PSO [33]: An iterative stochastic optimization technique based on swarm intelligence.

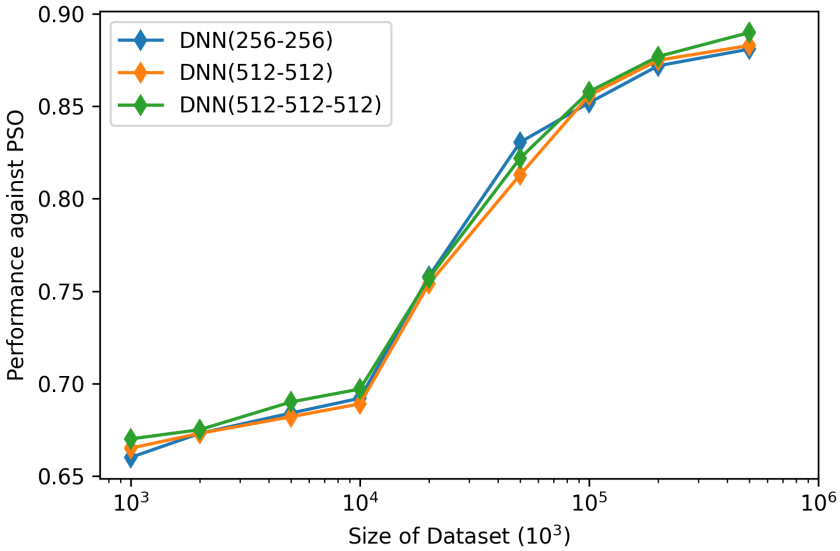

Figure 2.3: Percentage performance of PSO for models with different number of convolutional layers and hidden layer sizes on datasets of different sizes.

Table 2.1: System setting and DNN hyperparameters

Parameter	Value
Square area radius	500 m
BS transmit power	20 dBm
D2D Pairwise distance	2-50 m
Path loss exponent	η= 3.76
Noise PSD	-174 dBm/Hz
Channel bandwidth	5 MHz
Hidden layer size	(512-512-512-256)
Epoch size	30
Batch size	32
Learning rate	1×10^{-3}
Dropout rate	0.5
Optimizer	ADAM

2.4.2 Numerical Results

Figure 2.3 demonstrates the comparative performance of DNN with different structures in relation to the size of the training set. Either by adding more convolution layers or by parameterizing with larger MLPs, the performance of DNN could be boosted. To demonstrate the benefits of expanding MLPs, we show the results of DNNs with layer numbers (2 and 3) and the hidden size of MLPs (256 and 512). When 5×10^5

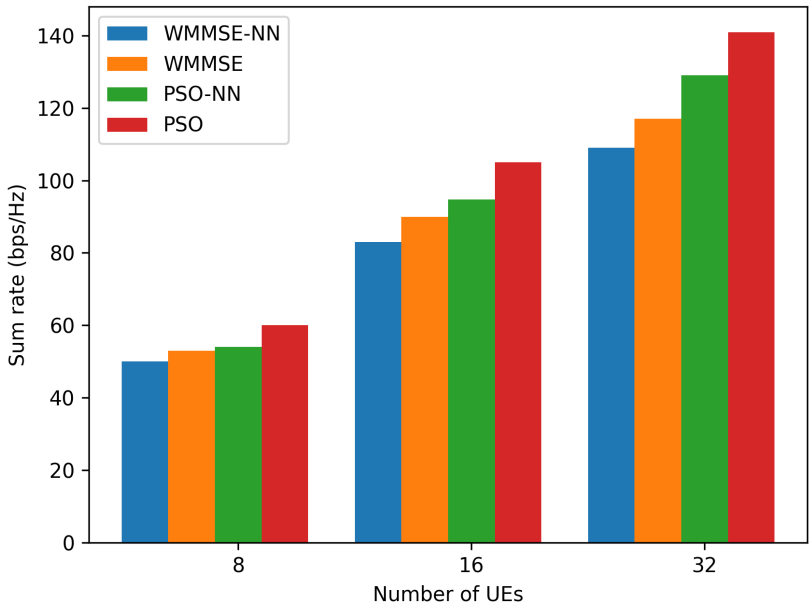

Figure 2.4: General sum rate for DNN with different UEs density.

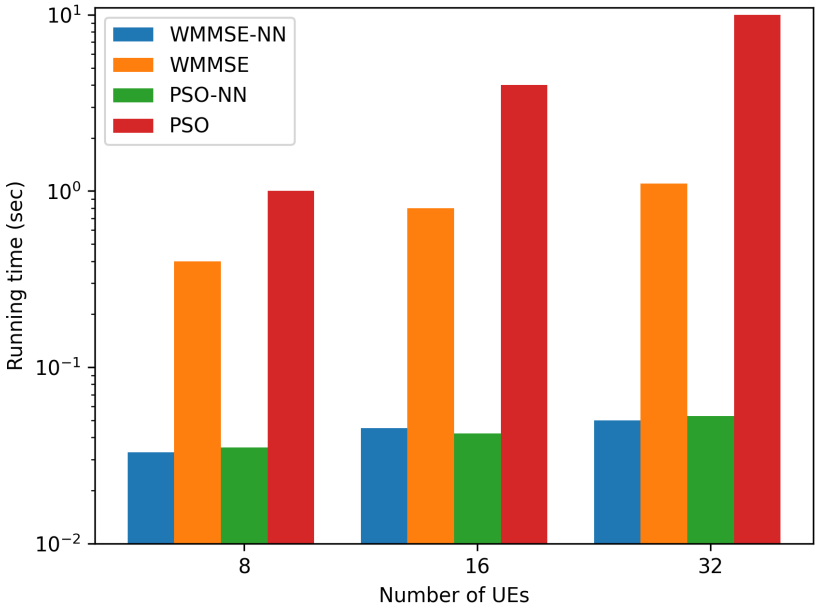

Figure 2.5: Average run time for 100 trials.

samples are input, the relative performance of a 3-layer DNN with a hidden size of 512 is 91.5 percent. Adding extra hidden layers to the DNN does not result in a substantial improvement; rather, it can easily lead to overfitting. In order to balance performance and complexity, we use this DNN structure in the subsequent trials.

Figure 2.4 and Figure 2.5 depict the sum-rate and runtime findings in relation to the number of UEs. As seen in Figure 2.4, the DNN-based approach outperforms its WMMSE counterpart as the number of UEs increases. The relative sum-rate performance of DNN compared to the optimal PSO algorithm is 91.5 percent.

Moreover, Figure 2.5 illustrates the runtime comparison between PSO, WMMSE-NN, and WMMSE for 1000 system states. AI inference is performed on an Nvidia 2070s development card using offline-trained DNN architecture. We observe that the proposed DNN approach outperforms the computationally intensive WMMSE technique by drastically lowering its execution time. In the case of $K = 12$ UEs, for example, WMMSE requires 1,056 seconds whereas DNN requires only 0.08 seconds. For iterative WMMSE, the running time increases dramatically with the size of the problem dimension. In contrast, DNN-based power allocation requires less processing time and is much less variable. For a trained DNN model, the number of computations is constant. The processing time fluctuation comes from the uncertainty of the computation of different floating point numbers and the read system time. For the heuristic algorithm, the time fluctuations mainly come from the different initializations, i.e., different search starting points can lead to significant differences in the time required to find the optimal solution.

2.5 Conclusion to Chapter 2

In this chapter, we've introduced a supervised learning-based framework, the PSO-DNN model, which is designed for solving resource allocation challenges in D2D networks, particularly in complex, real-world wireless communication scenarios. We've demonstrated the effectiveness of this solution within specific network setups and use cases, highlighting its ability to approximate heuristics. It's worth noting that these features, such as heuristics' approximations, are generally applicable to a broad range of networks and scenarios. DNNs have been shown to have the capacity to approximate virtually any function with adequate training, making this framework versatile.

Our results show that the trained PSO-DNN model outperforms the WMMSE-based

algorithm, especially in terms of computational time, while its performance is only slightly worse than that of the PSO algorithm (by 92%). This is a significant improvement, given that it saves 99% in computational time.

However, there are some challenges that we haven't fully addressed in this study, and further research is necessary. Determining the optimal DNN structure is a critical aspect, and this structure heavily depends on the specifics of the large-scale D2D communication system setup and the choice of hyperparameters. Additionally, DNNs don't efficiently utilize the communication network's topology, which can affect learning efficiency. In the next chapter, we explore optimal solution structures for more complex power allocation problems and integrating them into a deep learning-based framework.

Chapter 3

Resource allocation in Homogeneous D2D Network with Graph Learning in Supervised Manner

As discussed in chapter 2, DL models have found widespread application in various communication networks, proving highly effective for addressing a range of issues like network design, traffic prediction, and resource allocation. Nonetheless, many of these studies have only made partial use of the network's topology. This limitation arises because most deep neural networks are tailored for handling data structured in a Euclidean manner. In recent years, graph-based deep learning techniques, exemplified by GNNs, have emerged to address this gap, offering solutions for non-Euclidean structured data. GNNs have demonstrated a capacity to excel in solving problems within communication networks. Their strength lies in their ability to efficiently capture hidden spatial information within the network topology and generalize effectively even in cases where the network is dynamic or possesses invisible topologies. In this chapter, we introduce two PSO-based supervised graph neural learning frameworks.

3.1 Background Knowledge of Graph Learning

Graph is a foundational mathematical concept used to depict universal relationships, consisting of sets of nodes and the connections that link them. In recent times, graphs have found extensive use in representing a wide array of real-life scenarios, resulting in their broad acceptance as fundamental data structures. Within the realm of machine learning, approaches rooted in graph structures, commonly known as graph machine learning, involve methods like graph embeddings and neural networks [34].

Graph embedding refers to transforming the nodes or edges of a graph into a vector

space with reduced dimensions. This process is alternatively referred to as network embedding or graph representation learning [35]. In order to maintain the integrity of the graphs structure and characteristics, the voluminous, multi-dimensional, diverse, intricate, and ever-changing data is transformed into a standardized, reduced-dimensional, and compact vector representation. The related study aims to accomplish tasks such as node classification and clustering, link prediction, graph reconstruction, and visualization. The proposed technique is designed to have a low computational complexity. Hence, graph embedding is categorized as a preparatory step in real scenarios.

Graph Neural Network (GNN) is a neural network tailored for the analysis and processing of graph-based data [38]. GNN models are adept at handling input data characterized by diversity in scale, heterogeneity, and complex topological features. Their demonstrated capability to efficiently extract intricate topological information, identify critical and complex data aspects, and perform swift data processing is both compelling and reliable. The GNN framework includes a range of model types, including Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), Graph Autoencoders (GAE), and Graph Spatiotemporal Networks (GSTN).

In graph theory, a simple graph is defined as $G = (V, E)$, where V is the set of nodes and E represents the edges connecting these nodes. Based on the connectivity relationship, $\mathcal{N}(v_i)$ signifies the set of neighboring nodes of v_i, and each element in the degree matrix D is $D_{ii} = \mathcal{N}(v_i)$. The Laplacian matrix of the undirected graph is introduced and defined as $L = D - A$. Furthermore, the normalized Laplacian matrix is defined as $L = I_N - D^{-\frac{1}{2}} A D^{\frac{1}{2}}$, where N stands for the number of nodes, and I_N is the identity matrix with dimension N. The node feature matrix of the graph is expressed as $X \in R^{N \times d}$, where d represents the dimension of the node feature vector.

The graph embedding (GE) paradigm is introduced initially. In mathematics, embedding refers to a mapping function denoted as $f : X \to Y$, which facilitates the transfer of points from one space X to another space Y. Typically, embeddings are performed by mapping data from a high-dimensional abstract space to a lower-dimensional space. This practice is motivated by the fact that neural networks are more adept at processing data with lower dimensions, resulting in improved efficiency.

GCNs are inspired by the principles of convolutional neural network (CNN) models [36]. The primary objective in achieving the final neural network model entails the incorporation of variable parameters, followed by the utilization of gradient descent to optimize their performance. The spectral-based graph convolution method utilizes

concepts from signal processing to represent the attributes of nodes in a graph. The aforementioned technique can be classified as a convolutional transformation within the Fourier transform. The processing procedure involves the integration of the time domain signal with the Laplace eigenfunction.

The graph convolution operation, denoted as X_*, in GCN is precisely defined as follows:

$$X_* = W\left(I_N + D^{-\frac{1}{2}}AD^{\frac{1}{2}}\right)X, \tag{3.1}$$

where W denotes the learnable weight matrix. To address the concern of potential gradient explosion, an additional transformation is applied to the graph convolution process as

$$X_* = W\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}\right)X, \tag{3.2}$$

where $\tilde{A} = I_N + A$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

On the contrary, spatial-based GCNs establish convolution operations according to the graph's topology. For instance, the Message-Passing Neural Network (MPNN) suggests using a message-passing function, which encompasses a message-passing phase and a readout phase [39]. The message-passing phase is defined as follows:

$$m_{v_i}^t = \sum_{v_j \in \mathcal{N}(v_i)} \mathcal{M}^t\left(X_i^{t-1}, X_j^{t-1}, e_{ij}\right), \tag{3.3}$$

where $m_{v_i}^t$ denotes the message aggregated from the neighboring node of v_i, and $\mathcal{M}^t$ denotes the aggregation function, and t is the number of iteration. The message readout process is subsequently defined as follows:

$$X_i^t = u^t\left(X_i^{t-1}, m_{v_i}^t\right), \tag{3.4}$$

where denotes u^t the message readout function.

The graph attention network model presents a novel approach to handle graph-structured data by leveraging the attention mechanism [37]. Attention mechanism can selectively concentrate on the most crucial information amongst a vast quantity of data. The primary objective of Graph Attention Networks (GAT) is to calculate the latent representation of each node inside a graph by prioritizing the surrounding nodes in its vicinity by utilizing the attention mechanism. The proposed approach is a graph convolution technique that operates space-based, determining the weights of adjacent nodes based on their respective degrees. Therefore, it is suitable for inductive learning issues and has

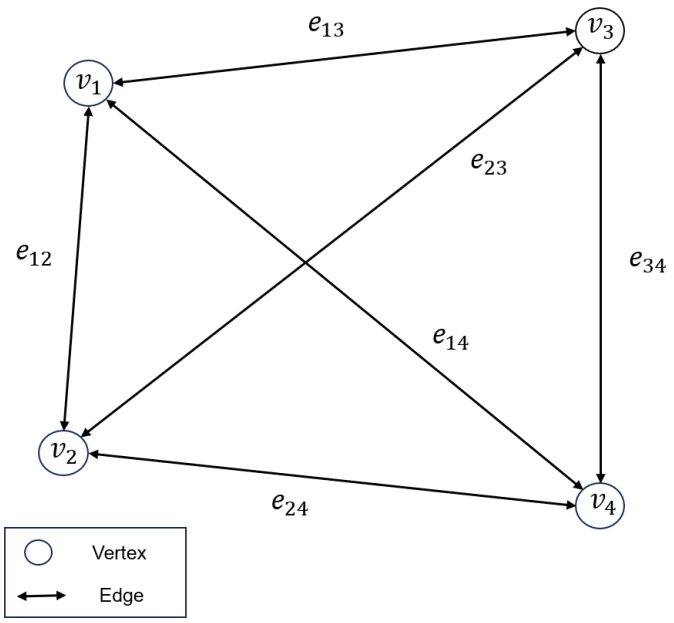

Figure 3.1: Example of D2D network's graph representation.

the potential to be applied to tasks involving hidden graphs. GAT integrates the attention mechanism into the propagation step and employs a multi-head attention mechanism to enhance the robustness of the model. The formulation of GAT is as follows:

$$X_i^t = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}(v_i)}\alpha^k\left(X_i^{t-1}, X_j^{t-1}\right)W^{t-1}X_j^{t-1}\right) \tag{3.5}$$

where σ denotes the activation function, and α^k denotes the k-th attention mechanism.

3.2 Graph Representation of D2D Networks

We formulate the sum rate optimization as learning over a direct homogeneous graph. Tuples $G = (V, E)$ provide a formal description of the network, where V and E represent the set of vertices and edges. The index of each communication link and interference link is denoted as a vertex $i \in V$ and an edge $(i, j) \in E$. A vertex i is the intersection of an entity and an edge (i, j) specifies a directed relationship between vertices i and j. The neighboring set of vertex i is denoted by $\mathcal{N}_i = \{j \in V \,|(j, i) \in E\}$. v_i and e_{ij} define the attributes of vertex i and edge ij, respectively. A simple network and its graph representation is shown in Figure 3.1.

Considering multiple types of vertices or edges in the heterogeneous graph. Denote the set of vertex types as S, the set of edge types as T. The graph can be represented by two mapping functions that map each vertex to its corresponding vertex type $\varphi : V \to S$

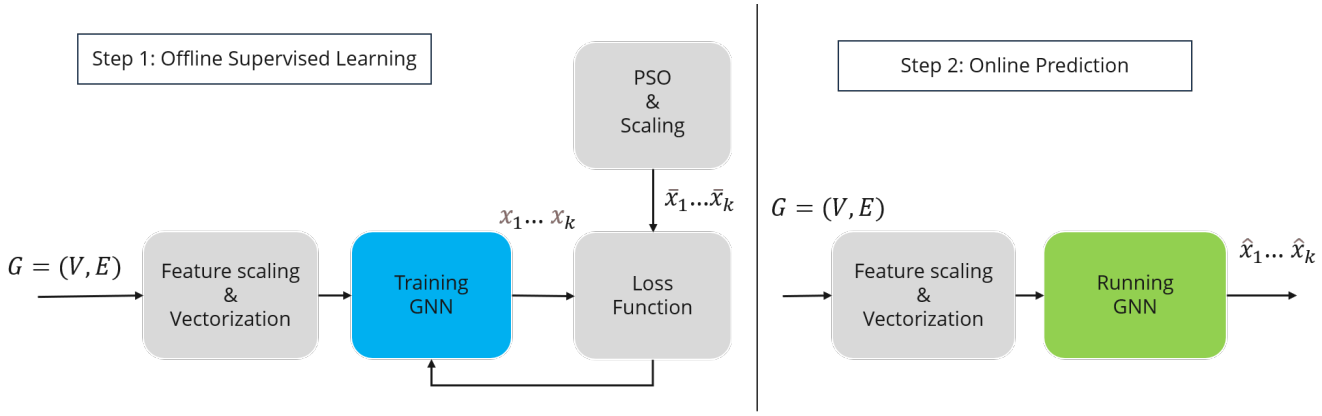

Figure 3.2: Architecture of the PA-GNN framework.

and each edge to its corresponding edge type $\psi : E \rightarrow T$. Denote the i-th vertex as v_i, $\mathcal{N}_i = \{j \in V \,|(j,i) \in E\}$ is the set of i's adjacent vertices. Node features are represented by $V \triangleq \{v_i\}_i$. Edge features are aggregated in $E \triangleq \{e_{ij}\}_{i,j}$, if edge (j,i) exists and 0 otherwise.

Each vertex's attributes contain its weight w_i, straight channel response h_{ii} and noise variance σ_i^2. Each vertex's attributes contain its channel response $e_{ij} = [h_{ij}, h_{ji}]$ from the interfering transmitters to the interfered receivers. Furthermore, because there are various connection kinds with unique numbers of send and receive antennas, the size of the graph features may change.

3.3 Architecture of Graph Learning Based Methods

The GNN-based framework trained in a supervised manner has two steps: (i) The first step searches through a heuristic algorithm for the labeled optimal assignment values and the corresponding network states as a dataset and trains the learning model. (ii) The second step runs the trained model in a real-time online application to predict the optimal assignment [45].

3.3.1 Learning Model 1: Message Passing Neural Network

GNNs have hierarchical neural network construction, like conventional MLPs. The MPNNs aggregate the features of the edges and adjacent nodes to update the representation of each node in each layer. The updating schema of the i-th node at layer l in GNNs is expressed as follows:

$$Aggregation : \alpha_v^{(l)} = \phi^{\alpha}\left(\left\{\beta_u^{(l-1)} : u \in \mathcal{N}(v)\right\}\right), \tag{3.6}$$

$$Combination : \beta_v^{(l)} = \phi^{\beta}\left(\beta_v^{(l-1)}, \alpha_v^{(l)}\right), \tag{3.7}$$

where $\alpha_v^{(l)}$ represent the feature aggregated by node v from its adjacent node at layer l. $\beta_v^{(l)}$ denote the feature of node v at layer l. ϕ^{α} is a parameterized function that encodes vertex and edge attributes for each edge (j, i). Then each vertex i aggregates the updates of edges. ϕ^{β} represents the combination function used to derive the vertex update by combining the aggregated edge update $\alpha_v^{(l)}$ with the vertex's current attributes $\beta_v^{(l-1)}$. Message transmission is completed when each vertex's knowledge is embedded in edge updates and consequently assimilated by its adjacent vertices.

With permutation invariance operations (e.g., sum, mean, and maximum), neighborhood aggregation can capture the permutation invariance attributes of the interference channel. Since the size of edge features varies with the number of antennas, features from different relations cannot be directly processed by ordinary GNN and these features should be treated separately. Therefore, we assign separate update functions to each relation using MLPs for parameterization. First, message transmission is performed in each relation. The target vertex then samples and aggregates partial updates from multiple relations to achieve its final update. Define the update at vertex i as:

$$e_{ij}^{(l)} = \phi^{e}\left(v_j^{(l-1)}, e_{ij}^{(0)}\right) \tag{3.8}$$

$$v_i^{(l)} = \phi^{v}\left(v_i^{(l-1)}, \max_{j \in \mathcal{N}_i} e_{ij}^{(l)}\right) \tag{3.9}$$

The initial edge attributes $e_{ji}^{(0)}$ are maintained in all edge update stages. This aids in maintaining performance stability. The attributes of each vertex are used as initial inputs in the forward calculation process, and then several iterations are performed to achieve the beamforming vector.

3.3.2 Learning Model 2: Graph Attention Networks

It is necessary to map these vectors to a high-dimensional space using the GAT layer to enhance the network topology information contained in the original low-dimensional features $\mathbf{n}_k$ obtained from the network. The input to the GAT network contains a set of node features denoted as $\mathbf{n} = (\mathbf{n}_1, \mathbf{n}_2, .., \mathbf{n}_K), \mathbf{n}_i \in R^{d_V}$, where each $\mathbf{n}_i$ is the feature of existing BS-to-UE pair v_i

Theoretically, GAT can use all nodes other than the central node to calculate the similarity with that central node. The dimensions of the node and edge feature spaces may vary. To enhance the expression of the node features, the layer of parameter-sharing neural networks is added to linearly transform the features (whether for nodes or edges), and the network is denoted by Att. Define $\mathbf{W} \in \mathbb{R}^{d_V \times d'_V}$ as the learnable weight matrix transforming input features linearly into high-level features, the dimension for feature vector of the node is made to change from d_V to d'_V. The self-attention mechanism calculates the similarity between the central node and the neighbor nodes, where a layer of the neural network calculates the similarity. The parameters are denoted by β^T, and the two transformed feature vectors are fed into this network after stitching. The importance coefficient of node v_i to node v_j is expressed as

$$\mathrm{c}_{i,j} = Attention\left(\mathbf{W}\mathrm{n}_i, \mathbf{W}\mathrm{n}_j\right). \tag{3.10}$$

We employ the mask attention technique to exclusively focus on the adjacent nodes of the first order, including the node itself. It computes the correlation between the nodes and their neighboring nodes and subsequently applies the softmax regularisation method. The coefficient can be formulated as

$$Attention\left(\mathrm{Wn}_i, \mathrm{Wn}_j\right) = \beta^T LeakyReLU\left(\beta^T\left[\mathrm{Wn}_i \parallel \mathrm{Wn}_j\right]\right). \tag{3.11}$$

Throughout this process, the features are jointly combined, with their respective parameters governed by the attention vector β^T and the activation function $LeakyReLU$. The resulting weight, which is normalized using the softmax function for node v_j is expressed as

$$\alpha_{i,j} = softmax_j\left(\mathrm{c}_{i,j}\right) = \left(\frac{\exp\left(\sigma\left(\beta^T\left[\mathbf{W}\mathrm{n}_i \| \mathbf{W}\mathrm{n}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\sigma\left(\beta^T\left[\mathbf{W}\mathrm{n}_{nk} \| \mathbf{W}\mathrm{n}_k\right]\right)\right)}\right), \tag{3.12}$$

where $\beta_n^T \in \mathbb{R}^{2d'_V}$ denotes the attention vector for node-based neighbors. The attention mechanism is implemented as a single-layer feedforward neural network, characterized by the parameters $\mathbf{W}$ and β. This neural network utilizes the $LeakyReLU$ nonlinearity with a negative slope 0.2. The variable $\alpha_{i,j}$ refers to the attention score that is used to quantify the significance of a neighboring node v_j concerning node v_i.

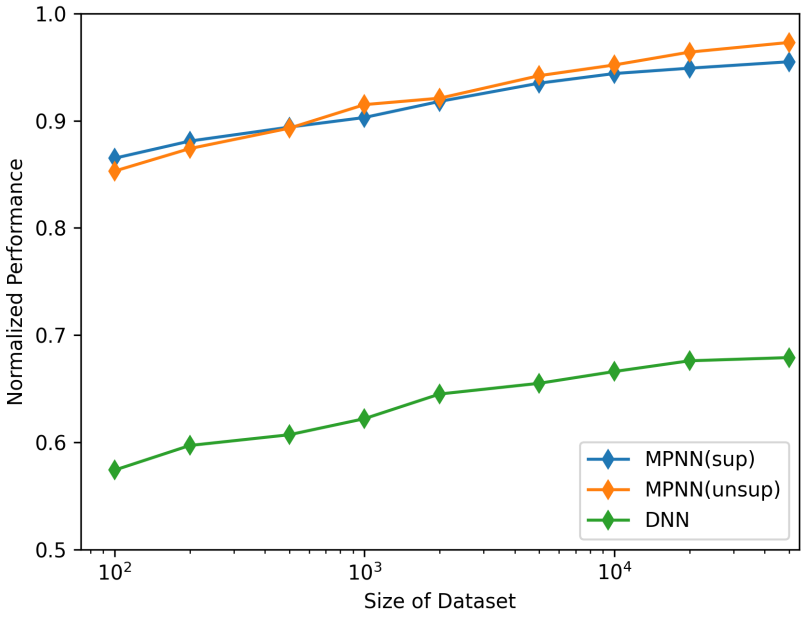

Figure 3.3: Performance versus size of dataset.

3.4 Simulation

In this section, the performance of PSO-MPNN and PSO-GAT are evaluated with different system parameters. We compare it with state-of-the-art baselines, including heterogeneous network embedding and graph neural network-based methods, to validate the proposed method's effectiveness.

3.4.1 Environment Setting

To assess the effectiveness of the proposed supervised Graph Neural Network (GNN), we employ the same homogeneous D2D network environment as described in Chapter 2. The computational device employed for conducting the simulations in this investigation is equipped with a central processing unit (CPU) of Intel(R) Core CPU i7-12700H, Graphics processing unit (GPU) of Nvidia rtx 2070s and a random access memory (RAM) capacity of 32 GB. The experimental methodology utilized the Deep Graph Library (DGL) and PyTorch framework as the foundation.

3.4.2 Numerical Results

The training efficiency of the learning-based approach is initially examined. As shown in Figure 3.3, the performance of the GNN gradually improves as the training sam-

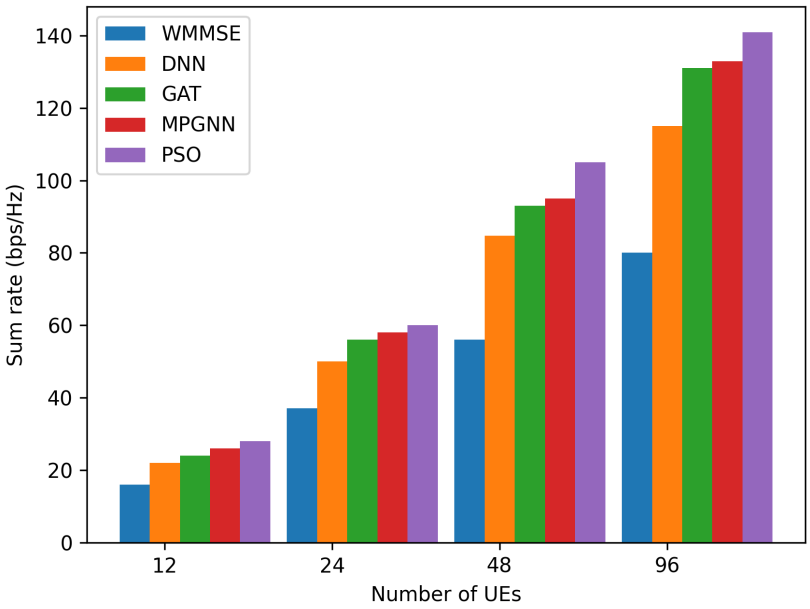

Figure 3.4: Performance versus number of D2D pairs.

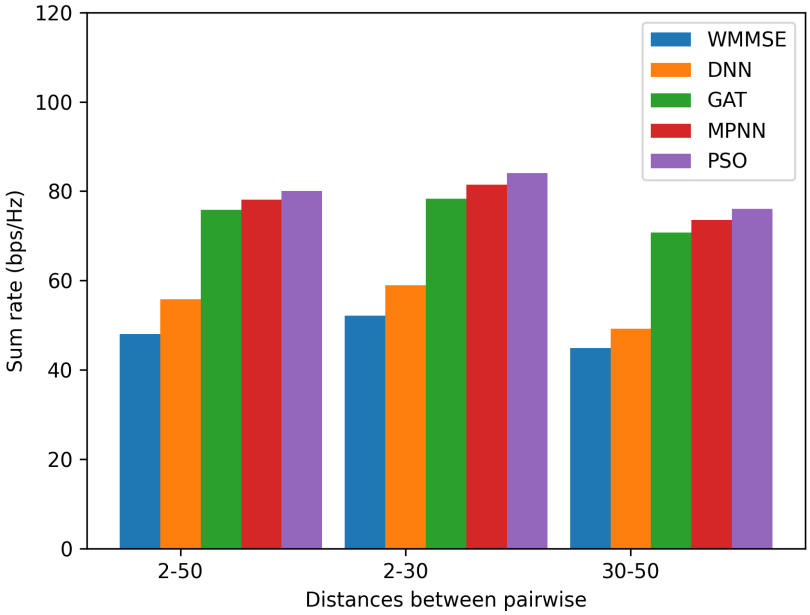

Figure 3.5: Performance versus pairwise distances.

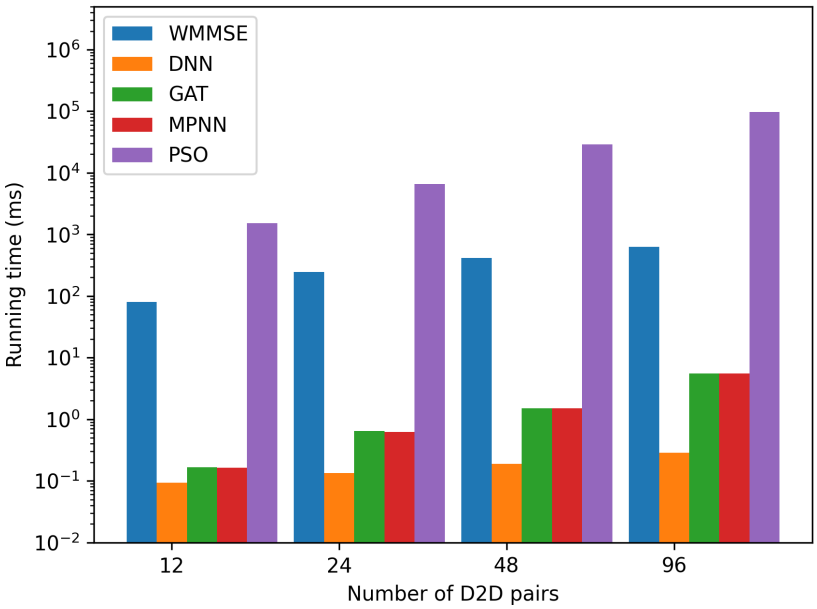

Figure 3.6: Execution time of different benchmarks.

ple size increases. Overall, there isn't a significant difference in performance between supervised and unsupervised trained GNNs, and supervised GNNs with fewer samples tend to perform better, possibly because our proposed method's input features include prior knowledge of the near-global optimal allocation. Furthermore, classical supervised Deep Neural Networks (DNNs) perform the worst among all the methods. Given the data-driven nature of supervised DNNs, they typically require large training datasets. Due to their superior sample efficiency, GNNs are often recommended for solving real-world problems in wireless networks, especially when collecting ample training data might be costly or inconvenient.

Next, we assess the scalability of the proposed method in networks with varying pairwise distances and densities. As shown in Figure 3.4, we examine the performance of the proposed method alongside benchmarks for different network sizes and link densities. When there are 12 pairs in the network, the performance of MPGNN and GAT is quite similar and differs from the near-global optimal solution of PSO by only 0.05. In the same environment, where the number of links is successively increased by factors of 2, 4, and 8, and the area of the region is similarly expanded, the normalization ratio of MPGNN remains consistently higher than 0.97, while that of GAT only stays at 0.95. This suggests that supervised GNN is not the best solution in this particular case. Possible reasons why MPGNN slightly outperforms GAT are that, in comparison to GAT which primarily focuses on the node features during training, MPGNN concurrently

aggregates edge weights as additional information within the nodes. This supplementation to the sample features contributes to its slightly superior performance compared to GAT.

In Figure 3.5, we can observe the performance of the GNN-based algorithm across different D2D pairing distances. Notably, the GNN-based algorithm consistently maintains an average normalization rate above 0.96 compared to the PSO algorithm, even as the distribution of pairing distances varies. This robust performance can be attributed to the GNN's ability to incorporate channel gain dependencies on distance by embedding geometric information of the wireless network into the node features. Consequently, the GNN-based optimization framework outperforms the other benchmark schemes across all three tested parameter settings.

As shown in Figure 3.6, PSO and WMMSE take up a lot of time as the number of D2D pairs increases and the problem becomes more complex as the network scales. Traditional RA algorithms are usually very time-consuming and unsuitable for real-time applications. In contrast, the optimization problem in D2D networks is greatly helped by the GNN-based approach. It is more than 100 times faster than standard algorithms and more than 1000 times faster than the heuristic algorithm PSO. This significant speedup of the proposed method augurs well for its real-time implementation on wireless networks. Since the proposed MPGNN method has the same network structure and input features as GAT, its runtime performance is comparable. DNN obtains better runtime performance than GNN due to ignoring graph features, but its terminal performance and sampling efficiency are poor.

3.5 Conclusion to Chapter 3

In this chapter, we have developed two scalable GNN-based neural network frameworks, namely MPGNN and GAT, to tackle the resource allocation problem in homogeneous D2D wireless networks. Our focus has been on crafting neural structures that meet crucial performance criteria such as minimal training cost, high computational efficiency, and strong generalizability, which are paramount in contemporary learning-based approaches.

We also seamlessly integrated heuristic and learning-based optimization techniques into our approach. In this paper, our proposition revolves around a GNN model that leverages supervised learning for parameter updates. This approach is distinctive from

the majority of other GNN methods, as it effectively utilizes node and edge features to illustrate the deep components' interactions, enhancing the model's expressiveness.

Our simulations have demonstrated that the proposed PSO-GNN method outperforms other DNN-based methods in maximizing the sum rate. This is primarily attributed to its effective utilization of a priori knowledge concerning the optimized solutions sought by heuristic algorithms. Furthermore, by observing the performance difference between MPGNN and GAT, we've gained insights into the influence of edge features on the models performance, which paves the way for our future research.

Chapter 4

Resource allocation in Heterogeneous D2D Network with Edge Feature Enhanced Graph Attention Network in Unsupervised Manner

Graph-based deep learning methods have emerged as a promising approach for addressing network resource allocation challenges. Given the inherently topological nature of mobile networks, leveraging GNNs proves advantageous in handling data structured as graphs. However, a predominant limitation of these methods lies in their emphasis on node features during the learning process, often neglecting or oversimplifying the role of edge features, which are equally vital. In this chapter, we introduce a novel design known as Heterogeneous Edge Feature Enhanced Graph Attention Network (HEGAT). This design directly connects the evolving network topology with the optimal resource allocation strategies during the learning phase. Through extensive simulations, we establish the exceptional performance and robust generalization capabilities of HEGAT in cellular networks characterized by diverse parameter settings.

4.1 Background Knowledge of Heterogeneous Graph

Previous chapters have underscored the necessity for resource allocation techniques to align more effectively with the underlying structure of communication networks. Conventional deep neural networks are explicitly designed for processing data organized in Euclidean structures, such as images and videos. In response to this limitation, the field has witnessed the emergence of GNNs, a class of graph-based deep learning models capable of handling non-Euclidean structured data, aiming to bridge this gap. While GNNs have made substantial progress in addressing resource allocation challenges within communication systems, they encounter two significant challenges. One

noteworthy observation is that the majority of prior research has predominantly emphasized node features while overlooking the potential value embedded in edge features. There is also a pressing need for a more comprehensive assessment of the significance of neighborhood structures and the evaluation of connections between edges and nodes. There are also some graph edge related studies as follows:

Message-Passing Neural Networks (MPNNs) contain multiple message-passing and readout phases. It is a paradigm that fully incorporates node features. At the same time, edge features are also used to characterize the network, and this feature updating mechanism has been proposed to predict node features [39]. Although MPNN adds edge information in the message-passing phase, its passing mechanism cannot learn the topological relationship between nodes and edges. Relational Graph Convolution Networks (RGCN) use forward-passing rules to add extra edge weights to the rule weight matrix [40]. However, experiments have shown that this simple aggregation is difficult to compute and does not significantly improve performance. Exploiting Edge Feature Graph Neural Network (EGNN) employs the aggregation function to combine node features while training separate attention weights for each dimension feature. The resulting dimensional outputs are then concatenated [44]. However, this phenomenon also leads to the loss of edge information.

On the other hand, real-world graphs often consist of various node and edge types and are collectively known as Heterogeneous Information Networks (HINs) [40]. Throughout this paper, we will refer to them as heterogeneous graphs for simplicity. Heterogeneity is a fundamental characteristic of these graphs, involving diverse node and edge types, each having distinct features residing in various feature spaces. These heterogeneous graphs, enriched with comprehensive information, find applications in numerous data mining tasks. Due to their intricacy, conventional graph neural networks cannot be straightforwardly employed to address the unique challenges posed by heterogeneous graphs. In light of these considerations, designing a graph neural network architecture with an attention mechanism for heterogeneous graphs must meet several new requirements. For example: The Heterogeneous Interference Graph Neural Network (HIGNN) is designed to handle heterogeneous network circumstances [13]. The communication connections between D2Ds serve as the nodes, and the nodes are classified according to the number of antennas on the transmitters in their links. In D2D downlink systems, this method is used to allocate power. The Heterogeneous Ultra-Dense Network (HUDN) is introduced to solve the resource allocation issue in communication scenarios with a

blend of D2D networks and cellular networks [45]. The communication links are regarded as nodes, and the nodes are categorized according to the types of devices they connect. Heterogeneous Graph Neural Network (HetGNN) is introduced to discover the power allocation policy for multi-cell multi-user systems. But these works did not consider the edge characteristics of the graph.

4.2 System Model and Graph Representation of Heterogeneous D2D Networks

4.2.1 System Model

This section considers a heterogeneous wireless network and simulates a downlink transmission scenario comprising numerous single-hop D2D pairs, as depicted in Figure 4.1. The main objective of this work is to optimize the resource management policy of each D2D pair in real time according to users' requirements. In contrast to prior research, the present study focuses on a practical scenario involving heterogeneous networks that incorporate multiple link types to establish communication connections. It is crucial to note that this scheme considers the presence of transmitters with varying numbers of antennas and receivers with a single antenna within the network.

A heterogeneous D2D network is considered with the transceiver pairs sharing the same spectrum of bandwidth B. Denote $\mathcal{M} \triangleq \{1, ..., M\}$ as the set of link types and $m, n \in \mathcal{M}$. The quantity indicating the number of transmit antennas for link type n is denoted as A_n. Let j_n index the j-th link of type n, $\mathbf{h}_{j_n j_n} \in \mathbb{C}^{A_n \times 1}$ denote the communication link channel between the transmitter and receiver of link i_m, $\mathbf{h}_{j_n i_m} \in \mathbb{C}^{A_m \times 1}$ denote the interference link channel from the transmitter of link i_m to receiver of link j_n. $\mathbf{x}_{j_n} \in \mathbb{C}^{A_n}$ represent the beamforming vector of link j_n. Then, the received signal at the receiver of the link j_n is given by

$$y_{j_n} = \mathbf{h}_{j_n j_n}^H \mathbf{x}_{j_n} s_{j_n} + \sum_{i_m \neq j_n} \mathbf{h}_{j_n i_m}^H \mathbf{x}_{i_m} s_{i_m} + z_{j_n}, \tag{4.1}$$

where $z_{j_n} \sim \mathcal{N}\left(0, \sigma_{j_n}^2\right)$ denote the additive white Gaussian noise (AWGN), $s_{j_n} \sim \mathcal{CN}\left(0, 1\right)$ denote the desired symbol of link j_n. Let $\mathbf{X} \triangleq \{\mathbf{x}_{j_n}\}_{j_n}$ denote the matrix of beamforming vector for all communication links. The data rate to a receiver of j_n is formulated as:

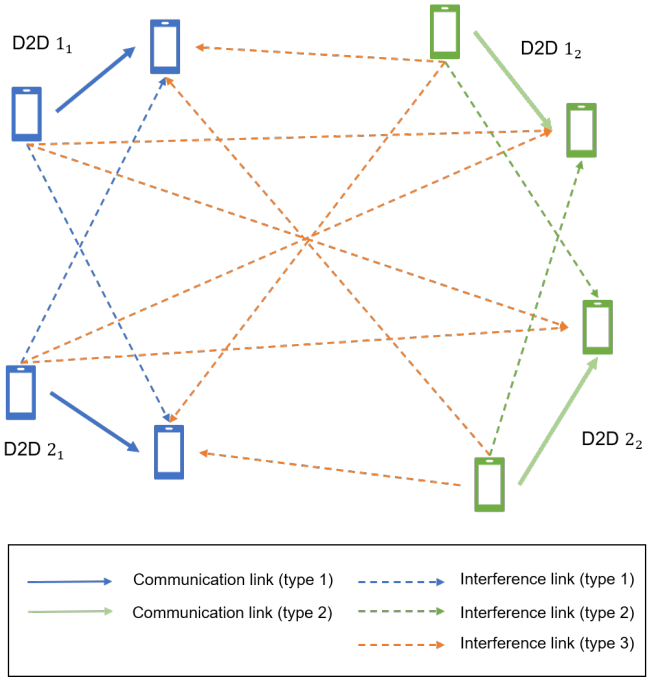

Figure 4.1: Graph modeling of D2D network with two types of links.

$$\xi_{j_n}(\mathbf{X}) = B\log\left(1+\frac{\left\|\mathbf{h}_{j_n j_n}^H \mathbf{x}_{j_n}\right\|_2^2}{\sum_{i_m\neq j_n}\left\|\mathbf{h}_{j_n i_m}^H \mathbf{x}_{i_m}\right\|_2^2+\sigma_{j_n}^2}\right). \tag{4.2}$$

The weighted sum rate is used to evaluate the system's performance. The object function for beamforming design to maximize the sum rate can be expressed as follows:

$$\begin{aligned}&\max_{\mathbf{X}}\sum_{j_n} w_{j_n}\xi_{j_n}(\mathbf{X})\\ &s.t.\ \|\mathbf{x}_{j_n}\|_2^2 \le p_{\max}\ ,\forall j,n,\end{aligned} \tag{4.3}$$

where p_{max} represents the maximum transmit power constraint of the communication link, and w_{im} denotes the weight of link i_m.

4.2.2 Graph Representation

Using a fully connected graph, we establish a representation of the D2D interference channel. The D2D pairs are regarded as nodes, while the interference links are regarded as edges. As shown in Figure 4.1, there are two types of nodes, depending on their communication type, and three types of edges, depending on the type of node they connect to.

The j_n-th transmitter-receiver pair is the j-th node of type n, denoted as v_{j_n}. The

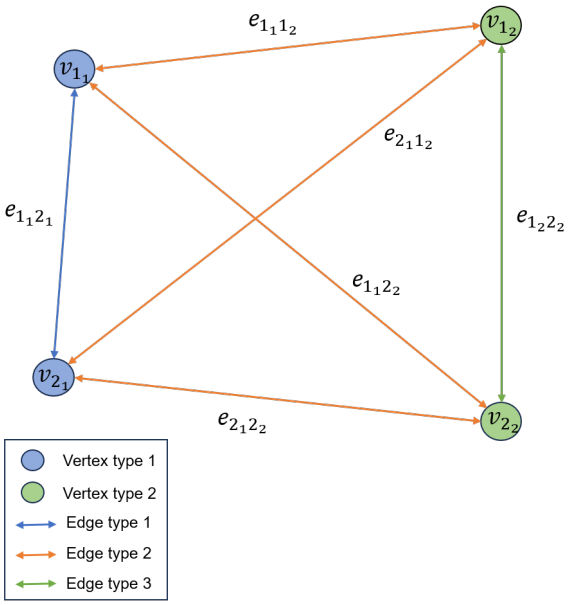

Figure 4.2: Graph representation.

node attribute contains the state of the direct channel, the weight, and the communication type. An edge $e_{j_n i_m}$ connecting two nodes, v_{j_n} and v_{i_m}, represents the interference link, where the attribute contains the states of the interference channels and types. In wireless networks, the attributes of both nodes and edges are as important as the network topology. To incorporate these attributes, we define the heterogeneous network graph as $G = (V, E)$, where V represents the set of nodes and $E \subset \{\{j_n, i_m\} | j_n, i_m \in V\}$ represents the set of edges. The mapping of nodes and edges to their respective features is denoted by $R : \phi \to \mathbb{C}^{d_V}$ and $S : \psi \to \mathbb{C}^{d_E}$, where d_V and d_E denote the dimension of feature space for node and edge, respectively.

4.3 Heterogeneous Graph Attention Networks

In this section, we present the framework of the proposed HNENN. The framework employs a hierarchical fusion of heterogeneous node-level and edge-level attention, allowing it to learn corresponding node and edge embeddings. The node and edge features are updated by aggregating information from the neighborhood.

4.3.1 Heterogeneous Transformation Process

In the heterogeneous graph, the node-based neighbor $\mathcal{N}_i$ of a given node i can be defined as the collection of nodes directly connected to node i. It should be noted that a given node's neighbors include the node itself. Similarly, the node-based neighbor $\mathcal{N}_{ij}$ of a

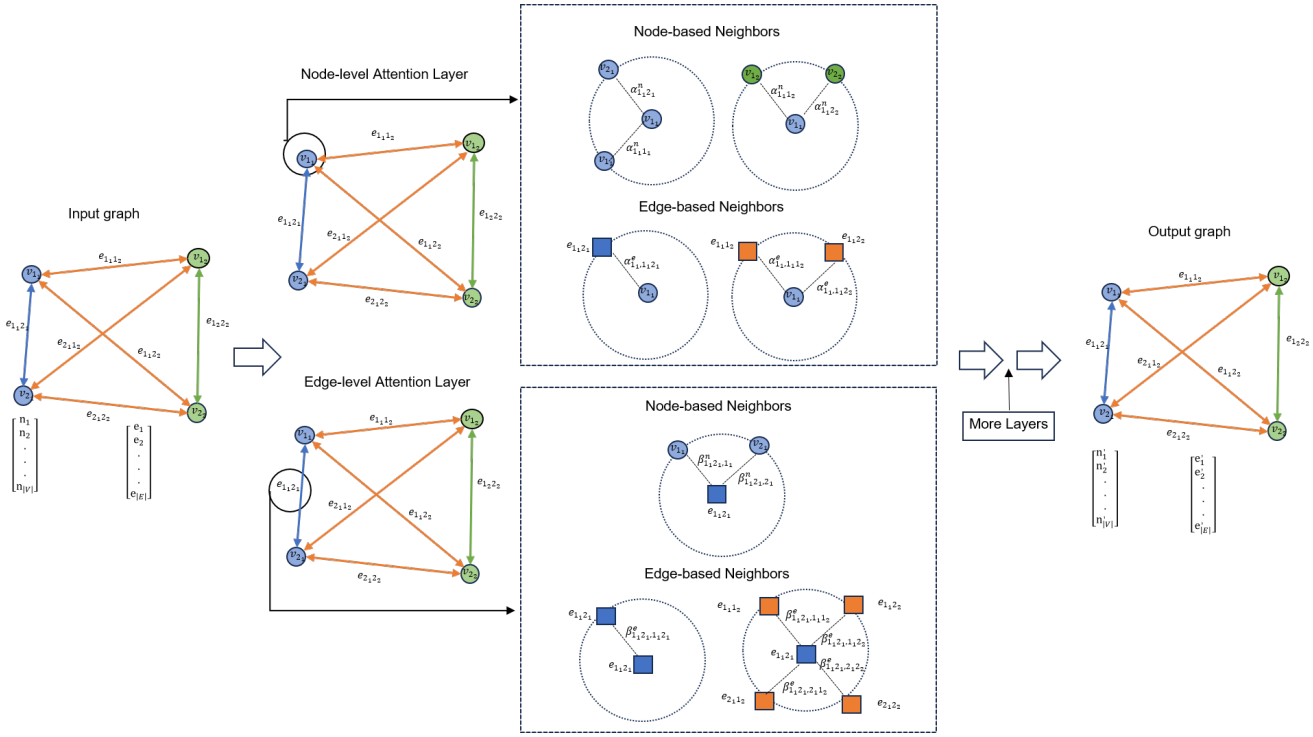

Figure 4.3: The architecture of the proposed HNENN for graph embedding. HNENN's layered attention mechanism, a node-level attention layer and an edge-level attention layer, alternate learning node, and edge embeddings.

given edge ij can be defined as the collection of nodes directly connected to edge ij.

As shown in Figure 4.2, circles represent nodes, and squares represent edges. In the node-level attention layer, the neighbors of node v_{1_1} based nodes are v_{1_1} and v_{2_1} of type 1, and v_{1_2} and v_{2_2} of type 2. In the edge-level attention layer, the neighbors of edge $e_{1_12_1}$ based nodes are v_{1_1} and v_{2_1} of type 1.

In the heterogeneous graph, the edge-based neighbor $\mathcal{E}_i$ of a given node i can be defined as the collection of edges directly connected to node i. Similarly, the edge-based neighbor $\mathcal{E}_{ij}$ of a given edge ij can be defined as the edges in the line graph directly connected to node edge ij. It should be noted that a given edge's neighbors include the edge itself.

As shown in Figure 4.2, in the node-level attention layer, the edge-based neighbors of node v_{1_1} are $e_{1_12_1}$ of type 1, $e_{1_11_2}$ and $e_{1_12_2}$ of type 2. In the edge-level attention layer, the edge $e_{1_12_1}$-based neighbors are $e_{1_12_1}$ of type 1, $e_{1_11_2}$, $e_{1_12_2}$, $e_{2_11_2}$ and $e_{2_12_2}$ of type 2.

In advance of aggregating information about the neighbors of nodes or edges in a communication network, it is essential to acknowledge that each node's or edge's neighbors may play distinct roles and exhibit varying importance in learning node or edge embeddings for resource allocation. We propose the incorporation of node-level and

edge-level attention, which effectively learns the significance of each node's or edge's neighbors within a heterogeneous graph. Using the proposed attention mechanism, we can combine the representations of these influential neighbors to build node or edge embeddings that show how they contribute to allocating resources.

Due to the heterogeneity of nodes and edges, different types of nodes or edges have varying feature space dimensions. The feature consists of attribute and type; the former are continuous complex variables, and the latter are discrete. We employ the type-specific transformation matrix M proposed in [89] for each node or edge type. This matrix facilitates the transformation of features from different types of nodes into a unified feature space. Denote n_i^{attr} as the vector of node i's attribute, and n_i^{type} as the vector of node i's type. The transformation process of node can be mathematically represented as $\mathrm{n}_i^{attr} = \mathrm{M}_\kappa \mathrm{n}_i^{attr}$. Note that we reused the n_i^{attr} to simplify the formulation. The feature of node i can be represented as $\mathrm{n}_i = \left[\mathrm{n}_i^{attr} \left\| \mathrm{n}_i^{type}\right.\right]$, which is obtained by concatenating its transformed node attribute and type information. Similarly, we can obtain the transformation process of edge as $\mathrm{e}_{ij}^{attr} = \mathrm{M}_\chi \mathrm{e}_{ij}^{attr}$, The feature of edge ij can be represented as $\mathrm{e}_{ij} = \left[\mathrm{e}_{ij}^{attr} \left\| \mathrm{e}_{ij}^{type}\right.\right]$. The node-level and edge-level attention mechanism can handle arbitrary types of nodes or edges using heterogeneous transformation operations.

4.3.2 Node-level Attention Layer

The node-level attention layer in HNENN is specifically designed to learn node embeddings by utilizing valuable information from heterogeneous edge features. It is evident that different neighbors of each node play distinct roles and have varying degrees of significance in generating the node embedding. To address this issue, we propose introducing a node-level attention mechanism. This mechanism determines neighbors' importance coefficients based on nodes and edges for node i.

In the l-th layer of HNENN, the input feature comprises a collection of node features denoted as $\mathbf{n} = \{\mathrm{n}_i \,|i \in [1, ..., |V|]\}$, where n_i represents the feature vector of node i. Additionally, there exists a collection of edge features denoted as $\mathbf{e} = \{\mathrm{e}_{ij} \,|i, j \in [1, ..., |V|]\}$, where e_{ij} corresponds to the feature of the edge pointing from node j to node i. The importance of node j or edge ij with respect to node i can be expressed as follows:

$$\mathrm{c}_{ij}^n = Att_{node}^n \left(W_n \mathrm{n}_i, W_n \mathrm{n}_j\right), \tag{4.4}$$

$$\mathrm{c}_{i,ij}^e = Att_{node}^e \left(W_n \mathrm{n}_i, W_e \mathrm{e}_{ij}\right), \tag{4.5}$$

where Att^{n}_{node} and Att^{e}_{node} represent the node-level attention performed by the deep neural network. For any given node i, the relationships formed with all its node-based neighbors share the Att_{node}. The weight of neighboring nodes or edges concerning a particular node depends on their features. Notably, the importance is asymmetric, meaning the importance of node i or edge ij to node j may not necessarily equal the importance of node j to node i or node i to edge ij. This observation highlights that node-level attention can preserve its asymmetric nature, which is an essential property in the context of heterogeneous graphs. Similarly, this property can be extended to edge-level attention. W_n and W_e are learnable weight matrices that transform the vertex features from low to high dimensions using linear mapping, a common feature augmentation method.

The incorporation of structural information is accomplished through masked attention, ensuring that embedding a specific node i is solely influenced by its neighboring nodes j or edges ij. Consequently, the importance coefficient of node j to node i is normalized using the softmax function as

$$\alpha^{n}_{ij} = softmax_j\left(\mathbf{c}^{n}_{ij}\right) = \left(\frac{\exp\left(\sigma\left(a^{T}_{n}\left[W_n\mathbf{n}_i \,\|W_n\mathbf{n}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_j}\exp\left(\sigma\left(a^{T}_{n}\left[W_n\mathbf{n}_i \,\|W_n\mathbf{n}_k\right]\right)\right)}\right), \tag{4.6}$$

where $a_n \in \mathbb{R}^{2d^{l+1}_{v}}$ denotes the node-level attention vector of node-based neighbors. It maps concatenated high-dimensional features to a real number. Specifically, this paper implements this mapping through a single-layer feedforward neural network. And σ denotes the LeakyReLU activation. The importance coefficient of edge ij to node i is normalized using the softmax function as

$$\alpha^{e}_{i,ij} = softmax_{ij}\left(\mathbf{c}^{e}_{i,ij}\right) = \left(\frac{\exp\left(\sigma\left(a^{T}_{e}\left[W_n\mathbf{n}_i \,\|W_e\mathbf{e}_{ij}\right]\right)\right)}{\sum_{st\in\mathcal{E}_i}\exp\left(\sigma\left(a^{T}_{e}\left[W_n\mathbf{n}_i \,\|W_e\mathbf{e}_{st}\right]\right)\right)}\right), \tag{4.7}$$

where $a_e \in \mathbb{R}^{d^{l+1}_{v}+d^{l+1}_{e}}$ denotes the node-level attention vector of edge-based neighbors.

Upon collecting the information regarding the significance coefficients α^{n}_{ij} and $\alpha^{e}_{i,ij}$, the aggregation process of embedding node i can be executed by considering the respective importance coefficients. Let $\mathbf{n}^{l+1}_{i}$ denote the feature output of node i for node-based neighbors. It can be expressed as a linear combination of the features,

$$\mathbf{n}^{l}_{i}\left[\mathcal{N}_i\right] = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha^{n}_{ij}W_n\mathbf{n}^{l}_{j}\right). \tag{4.8}$$

The heterogeneous nature of the graph data results in a high variance, mainly due to its scale-free characteristics. To enhance the stability of the self-attention learning process [90], we extend the node-level attention to multi-head attention [91]. This entails repeating the node-level attention for K times and connecting the learned embeddings to generate the following output feature representation:

$$\mathrm{n}_i^l\left[\mathcal{N}_i\right]=\sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^{n,k}W_n^k\mathrm{n}_j^l\right). \tag{4.9}$$

Similarly, the feature output of node i for edge-based neighbors can be expressed as:

$$\mathrm{n}_i^l\left[\mathcal{E}_i\right]=\sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{ij\in\mathcal{E}_i}\alpha_{i,ij}^{e,k}W_e^k\mathrm{e}_{ij}^l\right). \tag{4.10}$$

Consequently, the embedding of node i in layer $(l+1)$ is obtained by combining the edge-based neighbor embedding $\mathrm{n}_i^{l+1}\left[\mathcal{E}_i\right]$ and the node-based neighbor embedding $\mathrm{n}_i^{l+1}\left[\mathcal{N}_i\right]$ as:

$$\mathrm{n}_i^{l+1}=concat\left(\mathrm{n}_i^l\left[\mathcal{N}_i\right],\mathrm{n}_i^l\left[\mathcal{E}_i\right]\right) \tag{4.11}$$

where n_i^{l+1} is the updated embedding for node i with node-level attention mechanism at the l-th layer.

4.3.3 Edge-level Attention Layer

Edge features enhance the node embeddings within the node-level attention layer. Similarly, the edge-level attention layer utilizes the same approach to acquire the edge embeddings by fusion of node features. To update the edge embeddings, the initial step involves acquiring knowledge regarding the significance of both node and edge neighbors for each edge.

The importance of node j or edge ij with respect to node i can be expressed as follows:

$$\mathrm{c}_{ij,i}^n=Att_{edge}^n\left(W_e\mathrm{e}_{ij},W_n\mathrm{n}_i\right), \tag{4.12}$$

$$\mathrm{c}_{ij,ik}^e=Att_{edge}^e\left(W_e\mathrm{e}_{ij},W_e\mathrm{e}_{ik}\right), \tag{4.13}$$

The importance coefficient of edge ij to node i is normalized using the softmax function as

$$\beta_{ij,i}^{n} = softmax_i\left(\mathrm{c}_{ij,i}^{n}\right) = \left(\frac{\exp\left(\sigma\left(q_n^T\left[W_e\mathrm{e}_{ij} \,\|W_n\mathrm{n}_i\right]\right)\right)}{\sum_{k\in\mathcal{N}_{ij}}\exp\left(\sigma\left(q_n^T\left[W_e\mathrm{e}_{ij} \,\|W_n\mathrm{n}_k\right]\right)\right)}\right), \tag{4.14}$$

where $q_n \in \mathbb{R}^{d_v^{l+1}+d_e^{l+1}}$ denotes the edge-level attention vector of node-based neighbors. The importance coefficient of edge ij to edge ik is normalized using the softmax function as

$$\beta_{ij,ik}^{e} = softmax_{ij}\left(\mathrm{c}_{ij,ik}^{e}\right) = \left(\frac{\exp\left(\sigma\left(q_e^T\left[W_e\mathrm{e}_{ij} \,\|W_e\mathrm{e}_{ik}\right]\right)\right)}{\sum_{st\in\mathcal{E}_{ij}}\exp\left(\sigma\left(q_e^T\left[W_e\mathrm{e}_{ij} \,\|W_e\mathrm{e}_{st}\right]\right)\right)}\right), \tag{4.15}$$

where $q_e \in \mathbb{R}^{2d_e^{l+1}}$ denotes the edge-level attention vector of edge-based neighbors. Then, the feature output of edge ij for node-based neighbors can be expressed as

$$\mathrm{e}_{ij}^{l}\left[\mathcal{N}_{ij}\right] = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\mathcal{N}_{ij}}\beta_{ij,i}^{n,k}W_n^k\mathrm{n}_i^l\right). \tag{4.16}$$

The feature output of edge ij for node-based neighbors can be expressed as

$$\mathrm{e}_{ij}^{l}\left[\mathcal{E}_{ij}\right] = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{ik\in\mathcal{E}_{ij}}\beta_{ij,ik}^{e,k}W_e^k\mathrm{e}_{ik}^l\right). \tag{4.17}$$

Consequently, the embedding of edge ij in layer $(l+1)$ is obtained by combining the edge-based neighbor embedding $\mathrm{e}_{ij}^{l}\left[\mathcal{E}_{ij}\right]$ and the node-based neighbor embedding $\mathrm{e}_{ij}^{l}\left[\mathcal{N}_{ij}\right]$ as

$$\mathrm{e}_{ij}^{l+1} = concat\left(\mathrm{e}_{ij}^{l}\left[\mathcal{N}_{ij}\right], \mathrm{e}_{ij}^{l}\left[\mathcal{E}_{ij}\right]\right) \tag{4.18}$$

To facilitate a better understanding of the aggregation process involving the edge-level attention layer and node-level attention layer, a concise illustration is presented in Figure 4.3. The ultimate embedding is synthesized by aggregating both node and edge embeddings. Subsequently, this ultimate embedding can be utilized in various tasks, and different loss functions can be designed accordingly. The pseudocode of the proposed framework is shown in Algorithm 4.

4.3.4 Loss Function

During the forward computation, each node i takes its initial features $\mathrm{n}_i\left[0\right]$ as input and generates the embedding n_i^1. When layer $l > 1$, the node-level attention layer

Algorithm 4 Heterogeneous Node and Edge Graph Neural Network

Input: The heterogeneous graph $G = (V, E)$

The features of nodes and edges.

Output: The node-level attention weight α,

The edge-level attention weight β,

Final node embeddings n_i^l.

while $l \leqslant L$ **do**

for Each node $i \in V$ **do**

Find the node-based neighbors $\mathcal{N}_i$ and edge-based neighbors $\mathcal{E}_i$.

Do heterogeneous transformation $\mathrm{n}_i^{attr} = \mathrm{M}_\kappa \mathrm{n}_i^{attr}$.

Concate attributes and types $\mathrm{n}_j = \left[\mathrm{n}_i^{attr} \left\| \mathrm{n}_i^{type}\right.\right]$

Calculate the importance coefficient α_{ij}^n and$\alpha_{i,ij}^e$, obtain the embedding of node based neighbors $\mathrm{n}_i^l\left[\mathcal{N}_i\right]$ and edge based neighbors $\mathrm{n}_i^l\left[\mathcal{E}_i\right]$.

$\mathrm{n}_i^{l+1} = concat\left(\mathrm{n}_i^l\left[\mathcal{N}_i\right], \mathrm{n}_i^l\left[\mathcal{E}_i\right]\right)$

end for

for Each edge $ij \in E$ **do**

Find the node based neighbors $\mathcal{N}_{ij}$ and edge based neighbors $\mathcal{E}_{ij}$.

Do heterogeneous transformation $\mathrm{e}_{ij}^{attr} = \mathrm{M}_\chi \mathrm{e}_{ij}^{attr}$.

Concate attributes and types $\mathrm{e}_{ij} = \left[\mathrm{e}_{ij}^{attr} \left\| \mathrm{e}_{ij}^{type}\right.\right]$.

Calculate the importance coefficient $\beta_{ij,i}^n$ and $\beta_{ij,ik}^e$, obtain the embedding of node based neighbors $\mathrm{e}_{ij}^l\left[\mathcal{N}_{ij}\right]$ and edge based neighbors $\mathrm{e}_{ij}^l\left[\mathcal{E}_{ij}\right]$.

$\mathrm{e}_{ij}^{l+1} = concat\left(\mathrm{e}_{ij}^l\left[\mathcal{N}_{ij}\right], \mathrm{e}_{ij}^l\left[\mathcal{E}_{ij}\right]\right)$

end for

end while

Minimize the loss function $\mathcal{L}$.

Back propagation and update θ in HNENN

recursively generates updated features n_i^l by aggregaing n_i^{l-1} . Subsequently, n_i^{l-1} is passed to the output layer to obtain an estimate of the beamforming vector $\hat{\mathbf{x}}_{i_m} = \mathrm{n}_i^l$. The loss function $\mathcal{L}$ is defined as the negative expectation of the utility function across various channel realizations:

$$\mathcal{L}(\theta) = -E_H\left[\sum_{i,m} B\log\left(1 + \frac{\left\|h_{i_m i_m}^H \hat{\mathbf{x}}_{i_m}\right\|_2^2}{\sum_{j_n \neq i_m}\left\|h_{i_m j_n}^H \hat{\mathbf{x}}_{j_n}\right\|_2^2 + \sigma_{i_m}^2}\right)\right]. \tag{4.19}$$

The backpropagation technique is utilized to update the model parameters θ of the HNENN in an unsupervised manner based on the formulation above. As a consequence, a graph neural network model is obtained through training.

4.4 Simulation

In this section, the performance of HNENN is evaluated with different system parameters. We compare it with state-of-the-art baselines, including heterogeneous network embedding and graph neural network-based methods, to validate the proposed method's effectiveness.

4.4.1 Environment Setting

We conducted simulations on a heterogeneous D2D network. The network consists of two types of transmitters: one with a single transmit antenna and the other with two transmit antennas. Additionally, there are two link types, SISO (Single-Input Single-Output) and MISO (Multiple-Input Single-Output), with a ratio set at 2:1.

For the simulation setup, the transmitters of each D2D pair were randomly generated using a uniform distribution within a square region with a side length of D. Correspondingly, the receivers were uniformly distributed at a specified pairwise distance from the transmitters. The computation of the channel response is formulated as follows: $\mathrm{h}_{j_n i_m} = \sqrt{\beta_{j_n i_m}} g_{j_n i_m}$, where $\beta_{j_n i_m}$ is a real number, and $g_{j_n i_m}$ is a complex number representing the contributions of the large-scale fading and small-scale fading components, respectively. We employ a distance-dependent path loss model to capture the large-scale fading effect, while a Rayleigh fading model with a zero mean and unit variance represents the small-scale fading [32]. In particular, independent and identically distributed zero-mean complex Gaussian variables describe the small-scale decay. In the large-scale fading effect, the path loss and shadowing are measured using a model for scaled distance correlation. The channel response is a complex vector. Hence, it must be normalized to separate its real and imaginary components. After that, each of these elements is added individually to the GNN model.

The simulation experiments were executed using the TensorFlow framework with an NVIDIA RTX 2080 Ti GPU configuration. A 4-layer HNENN architecture was implemented. We allocate the same training, validation, and test sets for all baselines to ensure fairness. The datasets for all molecular networks were divided into training, validation, and test sets in an 8:1:1 ratio. The model is executed three times, and each experiment's average performance is recorded. System settings and GNN hyperparameters are summarized in Table 4.1.

Table 4.1: System setting and GNN hyperparameters

Parameter	Value
Area length D	500
D2D pair distance $[d_{\min}, d_{\max}]$	$[2\text{ m}, 5\text{ m}]$
Transmit power range $[p_{\min}, p_{\max}]$	$[10\text{ dBm}, 30\text{ dBm}]$
Pass loss model	$148 + 40\log_{10}(d\,[\text{km}])$
Learning rate	0.0001
Number of layers of GNN	4
Optimizer	ADAM
Batch size	64
Epoch count	300

4.4.2 Benchmarks

We compare the proposed method with several graph representation learning methods: the MPGNN, NENN, Random Edge Graph Neural Network (REGNN), and Random Strategy (RAND).

- MPGNN (2021) [86]: The presented approach represents a state-of-the-art learning-based method that relies on the message passing mechanism and is specifically designed for heterogeneous networks.
- NENN (2020) [73]:The proposed approach utilizes a hierarchical mechanism, incorporating both node-level and edge-level attention, to enhance the embeddings of nodes and edges across multiple neural network layers. It is specifically designed for homogeneous networks.
- REGNN (2021) [42]: It uses special REGNN architectures and forward-passing nonlinear graph convolution to combine spatial weights with channel coefficients. This makes it adaptable to networks with different types of connections.
- RAND: It selects variables within the constraints according to the uniform distribution.

4.4.3 Numerical Results

Comparsion with Different Network Scale

To compare the performance of different algorithms across a wider range, we set the parameters within the range of [2m, 50m] with 1600 training samples. While keeping

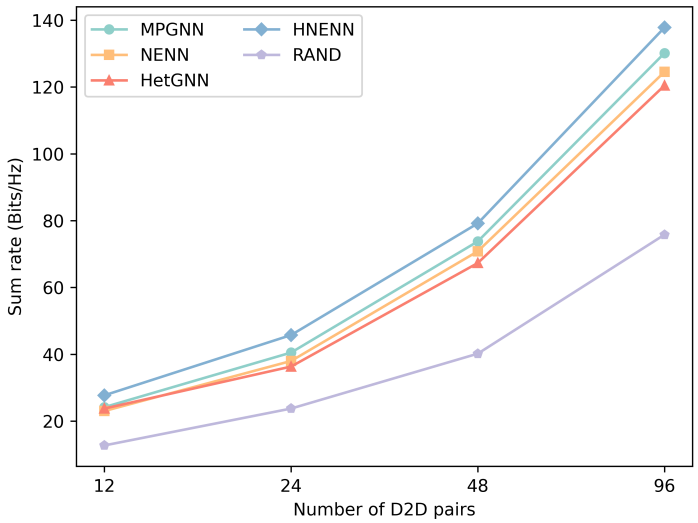

Figure 4.4: Performance comparisons with expanding spatial areas.

the link density constant, each doubling of the number of links results in the square region's sides expanding by a factor of p2 of the previous length. As depicted in Figure 4.4, the sum rate achieved by HNENN consistently outperforms other methods as the region expands. Specifically, when the number of links increases to 12, 24, 48, and 96, HNENN demonstrates a superior sum rate, surpassing state-of-the-art MPGNN by 8.9%, 8.4%, 7.3%, and 6.8%, respectively.

Subsequently, we assess the performance of different algorithms under varying link densities, with the number of links exponentially increasing while maintaining a fixed region side length. As shown in Figure 4.5, all algorithms exhibit a decline in link performance compared to the results in Figure 4.4. This is attributed to links being more closely spaced in congested environments, leading to more severe channel interference. When the number of links increases to 12, 24, 48, and 96, the performance improvement of HNENN relative to MPGNN is 4.2%, 3.8%, 4.5%, and 5.7%, respectively, and the performance gap widens with the increase in link density.

Comparsion with Different Network Parameters

Figure 4.6 illustrates the performance comparison results for 48 D2D pairs with different numbers of training samples. When the number of training samples is 200, 400, 800, and 1600, the performance of HNENN is improved by 1.8%, 2.4%, 3.2%, and 4.5% compared to MPGNN. Because HNENN is designed for heterogeneous graphs, it additionally considers the features of neighboring nodes and edges as embedded fea-

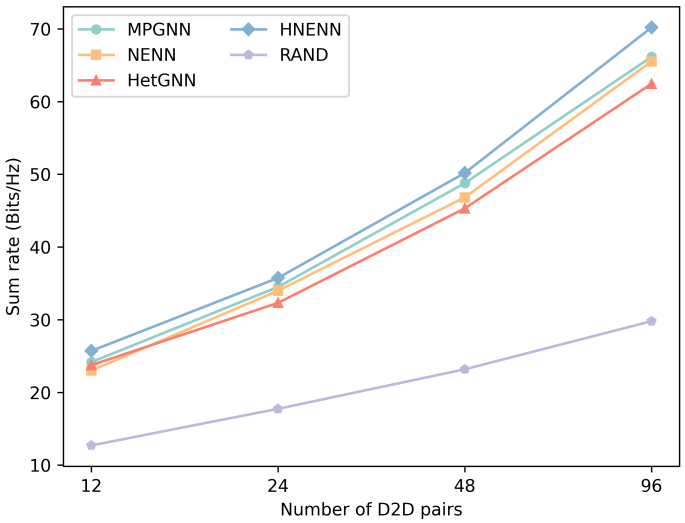

Figure 4.5: Performance comparisons with higher link density.

tures in the feature aggregation. Such high sample-efficiency features are very relevant for practical problems in wireless networks because the computational cost of obtaining enough training samples is expensive. Furthermore, HNENN can obtain high performance with fewer training samples, which has the potential for practical applications.

Figure 4.7 compares the performance between HNENN and benchmark methods under various D2D pairing distances. Notably, even when the distribution of D2D communication distances varies, the performance of HNENN consistently outperforms the benchmark schemes. This can be attributed to the heavy dependence of channel gain on distance, where shorter distances reduce the channel gain's attenuation. Consequently, HNENN's performance advantage is maintained across different D2D pairing distances.

Running Time Comparsion

As shown in Figure 4.8, the proposed HNENN has the same order of magnitude runtime as other benchmark algorithms because they share similar network structures and input features. However, HNENN consumes more time than other benchmark methods because the former requires extra time to acquire and aggregate the embedding features of nodes and edges, where each node iteratively updates the embedding features of itself and all its neighboring nodes and edges. When the number of links is 96, the running time of HNENN is 0.002s, much less than the required time for decision-making in wireless systems (0.02s), and satisfies the need for real-time decision-making. Therefore the proposed method is effective for real-time implementation in wireless networks.

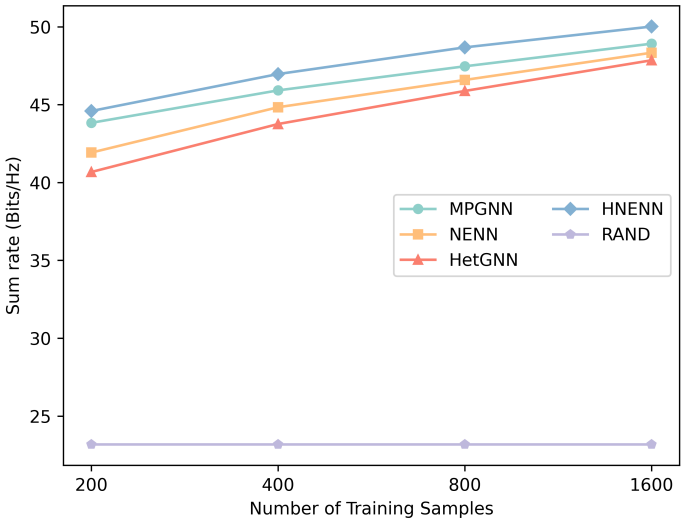

Figure 4.6: Performance comparisons with various number of training samples.

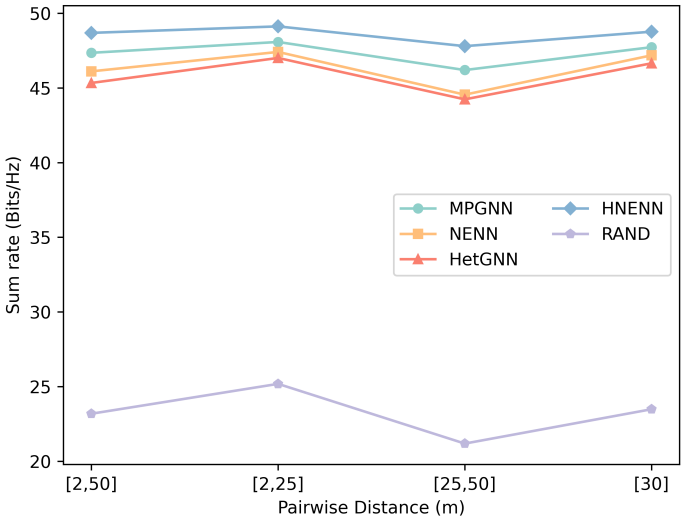

Figure 4.7: Performance comparisons with various pairwise distances.

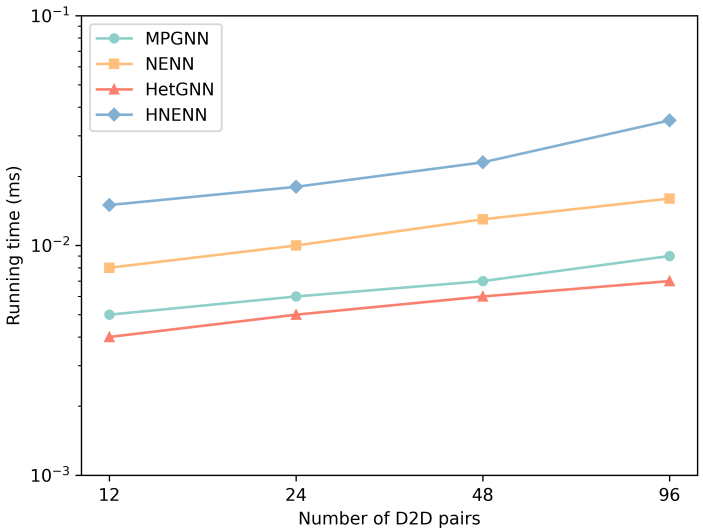

Figure 4.8: Running time comparison with various numbers of D2D links.

4.5 Conclusion to Chapter 4

This chapter presents a formulation of the wireless network resource management problem on a heterogeneous graph. The graph is a mapping function that relates node and edge features to variables associated with the resource management problem. These variables, whose values are yet to be determined, are defined on the nodes and edges of the graph. We propose the HNENN framework to address the challenges of heterogeneity. The HNENN model effectively integrates node and edge features, significantly enhancing the embedding of both nodes and edges across multiple neural network layers. It employs a sequential arrangement of node-level and edge-level attention layers to capture the importance of neighboring nodes and edges. Simulation results demonstrate that the proposed framework achieves superior sum rates. Additionally, the efficacy and robustness of this framework have been validated across diverse scenarios.

Chapter 5

Mean Field Reinforcement Learning for Optimal Resource Allocation in Heterogeneous D2D Network

Heterogeneous networks and D2D communication technologies have emerged as promising solutions for handling the growing mobile traffic [46]. However, the exponential growth of the state and action space in heterogeneous D2D networks renders traditional learning approaches impractical [49]. In this chapter, we propose an enhanced approach that combines Multi-agent Reinforcement Learning (MARL) with Mean-field Type Game (MFTG) theory to model approximate interactions between different classes of users in heterogeneous D2D networks [82]. Through extensive experimentation, we demonstrate that our proposed multitype Mean-field Double Deep Q Network (MTMF-Q) approach outperforms benchmark methods in terms of performance, highlighting its potential for practical implementation in ultra-dense communication network scenarios.

5.1 Background Knowledge of Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) techniques, particularly MARL, have gained attention as effective approaches for developing artificial intelligence (AI)-enabled solutions to wireless network resource allocation problems [47]. The interaction between a network entity acting as an agent and a dynamic and unpredictable network environment is often modeled as a partially observable Markov Decision Process (POMDP) [59] to learn optimal selection strategies. Huang et al. [58] proposed a cost-aware collaborative task execution scheme based on Multi-agent Deep Deterministic Policy Gradients (MADDPG) for energy-harvesting D2D networks with limited computing, network, and battery capacity. This scheme facilitates collaboration among multiple

energy-harvesting mobile devices to minimize task execution. Guo et al. [62] addressed the credit distribution issue using multi-agent proximal policy optimization (MAPPO), which marginalizes the actions of the current user equipment (UE) while keeping the actions of other UEs fixed. To ensure robust and stable learning, Mseddi et al. [61] proposed a collaborative approach based on QMIX [63], which employs hybrid networks conforming to monotonicity constraints. These studies highlight the use of DRL and MARL techniques in addressing resource allocation challenges in wireless networks, showcasing their potential for achieving efficient and collaborative solutions.

Managing scenarios with a large number of agents presents challenges as classical MARL techniques are limited in handling more than a dozen or tens of agents [76]. The scalability of multi-agent systems encounters two significant obstacles: The number of interactions between agents grows proportionally to the square of the number of agents. This leads to increased computational complexity and makes policy learning more challenging. The strength of interactions varies among agents and changes over time, posing difficulties in achieving convergence of the loss function [80]. These obstacles highlight the complexities involved in scaling up the number of agents in MARL systems and emphasize the need for novel approaches and techniques to address these challenges effectively [79].

Mean-field Game (MFG) theory is a practical approach for addressing problems involving a large number of agents. In such scenarios, each agent interacts with the environment by considering its interactions with the collective behavior of the other agents [65]. This approach significantly reduces the computational complexity associated with analyzing multi-agent systems. The integration of MFG and MARL shows great potential for applications in the field of communication. Ye et al. [48] utilized DRL techniques to train spiking neural networks (SNNs). They combined mean-field multi-agent reinforcement learning (MFRL) and Spiking Proximal Policy Optimization (S-PPO) to facilitate channel selection and power control. Yang et al. [66] proposed User Cluster Matching (UCM) that leverages variations in the user channel gain algorithm. They addressed the resource allocation problem in densely deployed user scenarios using the Mean-field Deep Deterministic Policy Gradient (MFDDPG) method. Chen et al. [67] introduced the Mean field Trust Domain Policy Optimization (MFTRPO) approach, which optimizes trust domain policies and neural network feature embedding methods for energy allocation to maximize communication efficiency. It is crucial to note that the use of MFG theory assumes agent homogeneity, implying that the properties of

agents should be similar. However, in real network environments, multiple types of agents with different attributes are often encountered [81].

To address the limitations of MFG and accommodate heterogeneous agents, a more adaptable game structure called MFTG has been proposed. In MFTG, agents are not required to be homogeneous, and interactions between agents and the environment are formulated by considering the interactions between collective behaviors within the same category and across different categories. In the field of communication, MFTG has been explored in various studies. For example, Zhang et al. [68] propose a heterogeneous MFMARL framework for routing resource allocation to optimize the communication energy efficiency of the Space-air-ground Integrated Network (SAGI-Net). Different types of devices, such as ground devices and Unmanned Aerial Vehicles (UAVs), are treated as heterogeneous agents in the framework. Li et al. [69] address the power control and trajectory design problem by modeling it as a discrete MFTG. They use a clustering algorithm to classify devices into different categories and propose a Mean-field Q Learning (MF-Q) algorithm to solve the joint optimization problem. Overall, the MFTG model is more suitable for tackling optimization problems in heterogeneous networks, where agents have diverse characteristics and behaviors.

5.2 System Model of Heterogeneous D2D Network

We investigate a downlink D2D wireless network featuring multiple types of communication links, each potentially exhibiting diverse attributes within heterogeneous environments. In this scenario, we assume that each receiver is equipped with a single antenna. However, it's important to note that the number of transmit antennas may vary across different links. All connections within the system follow a MISO configuration. It's noteworthy that the exact number of multiple antennas is not standardized and can vary between different links.

In a square region with an edge length of d, where each D2D pair is randomly distributed, all pairs share the same spectrum denoted as B. We describe the link types with $\mathcal{M} \triangleq \{1, ..., M\}$, where each type m has a different number of transmit antennas represented by A_m. Each link within the system is indexed as i_m for the i-th link of type m, with $i_m \in \mathcal{I}$ and the total number of links denoted as I. The channel responses, $h_{i_m i_m}$, signify the communication channel characteristics between the transmitter and receiver of link i_m, while $h_{i_m j_n}$ represents the interference channel response from the transmitter

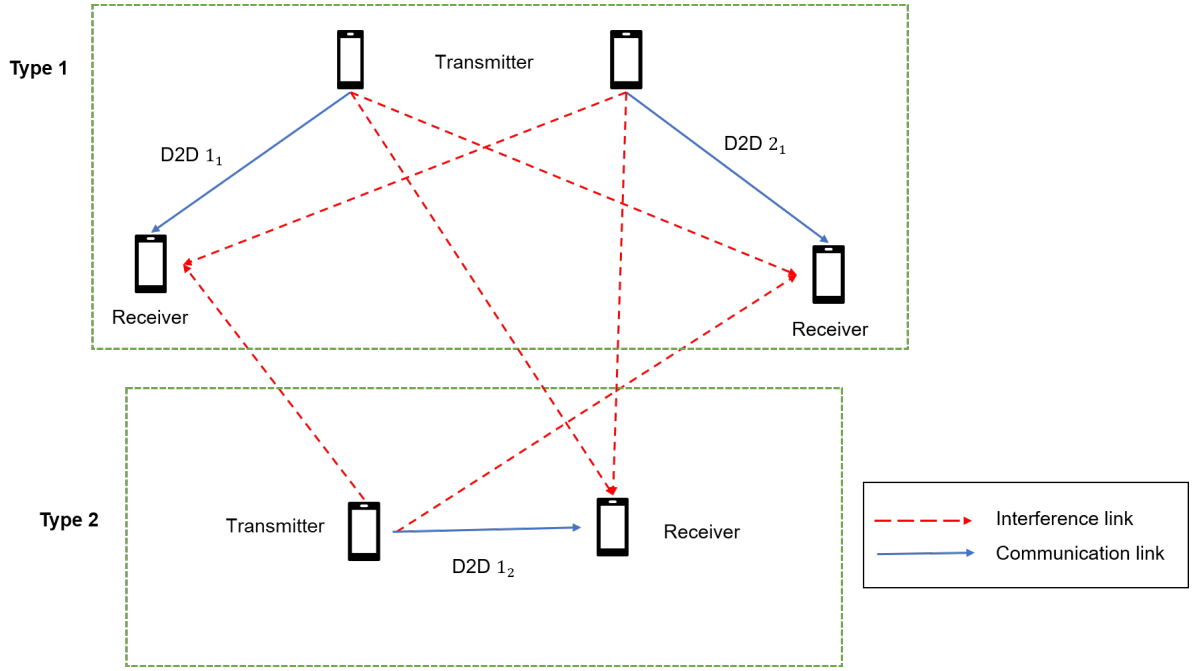

Figure 5.1: An instance of a heterogeneous D2D network with two distinct types of links.

of link j_n to the receiver of link i_m. These channel responses are of paramount importance in determining communication quality and interference levels within the network. At time t, the independent channel coefficient of the receiver for link i_m is expressed as:

$$\begin{aligned} y_{i_m}(t) =& h^H_{i_m i_m}(t)\, w_{i_m}(t)\, \sqrt{p_{i_m}(t)} u_{i_m} \\ &+ \sum_{j_n \neq i_m} h^H_{i_m j_n}(t)\, w_{j_n}(t)\, \sqrt{p_{i_m}(t)} u_{j_n} + z_{i_m}, \end{aligned} \tag{5.1}$$

where $z_{i_m} \sim \mathcal{N}\left(0, \sigma {i_m}^2\right)$ represents the additive white Gaussian noise (AWGN) ui_m is the signal symbol with zero mean and unit variance. The beamforming vector used by the transmitter of link i_m to send the signal $u_{i_m} \in \mathbb{C}$ to the receiver is denoted as $v_{i_m} \in \mathbb{C}^{A_m}$. p_{i_m} represents the transmit power of link i_m, and we denote the set of transmit powers as $P = \{p_{i_m}\}_{i_m}$. The normalized beamforming vector is w_{i_m}, and we denote the set of these normalized vectors as $W = \{w_{i_m}\}\, i_m$. The beamforming vector can be expressed as $v_{i_m} = w_{i_m}\sqrt{p_{i_m}}$. At time slot t, the data rate to receiver i of type m is formulated as:

$$\xi_{i_m}\left(W\left(t\right),P\left(t\right)\right)= \\ B\log\left(1+\frac{\left\|h_{i_m i_m}^{H}\left(t\right)w_{i_m}\left(t\right)\sqrt{p_{i_m}\left(t\right)}u_{i_m}\right\|_2^2}{\sum_{j_n\neq i_m}\left\|h_{i_m j_n}^{H}\left(t\right)w_{i_m}\left(t\right)\sqrt{p_{j_n}\left(t\right)}u_{j_n}\right\|_2^2+\sigma_{i_m}^2}\right), \tag{5.2}$$

Especially in cases where the connection type is single-input-single-output (SISO), the beamforming design simplifies into a power control problem. Here, p_{max} represents the power constraint. Over a time period T, the objective function of the optimization problem can be calculated as:

$$\begin{aligned}\max_{w_{i_m}(t),p_{i_m}(t)} \quad & \frac{1}{T}\sum_{t=1}^{T}\sum_{i_m}\xi_{i_m}\left(t\right)\left(w_{i_m}\left(t\right),p_{i_m}\left(t\right)\right) \\ s.t. \quad & \left\|w_{i_m}\sqrt{p_{i_m}}\right\|_2^2\leq p_{\max}\ ,\forall\ i\in\mathcal{I},m\in\mathcal{M}.\end{aligned} \tag{5.3}$$

Our primary research objective is to optimize the beamforming and power allocation policy to achieve an optimal sum rate in a D2D heterogeneous network. The sum rate signifies the total achievable data rate in the network. To ensure a fair comparison and analysis, we normalize the data rate by dividing it by the channel bandwidth, resulting in a unit of bits per second per hertz. Solving this optimization problem is crucial for enhancing the overall performance and efficiency of the network.

5.3 Problem Formulation of Resource Allocation Problem

The optimization problem modeled in the previous section is an integer nonlinear programming problem, which is a non-deterministic polynomial (NP) problem that is difficult to optimize in terms of polynomial time complexity approaches

5.3.1 Partially Observable Markov Decision Processes

In the decentralized architecture, multiple single-agent systems are integrated to create a multi-agent system. As the system's scale increases, information sharing can lead to significant signaling congestion, rendering it impractical. We employ a MARL system with Partially Observable Markov Decision Processes (POMDP) to model spectrum and power allocation problems, aiming to maximize the sum rate for the D2D communication network. A POMDP with I agents can be described using the tuple:

$\left(\mathcal{S}, \mathcal{A}^1, ..., \mathcal{A}^I, r^1, ..., r^I, \mathcal{P}, \gamma\right)$, where $\mathcal{S}$ denote state space which is a finite set of states, $\mathcal{A}^i, i \in \{1, ..., I\}$ is action space of agent i, r^i is reward function of agent i, $\mathcal{P}$ is the transition distribution.

In this system, every D2D pair functions as an independent learning agent, each with its decision-making strategy. These agents engage in ongoing interactions with the dynamic environment, which furnishes feedback concerning the performance of the agents' policies based on their historical actions. The state of the environment's evolution and the reward function acquired by each agent are contingent on the collective actions of all agents. Consequently, each agent must factor in not only the environment but also the interactions with other learning agents when adapting and optimizing its strategy.

5.3.2 Environment State

The concept of environment state is a comprehensive description of the system's state, and it fully captures the system's condition at any time step. The environment state in POMDP is a whole, unrestricted state containing every possible state. However, in POMDP, D2D must estimate or infer the present environmental state from observation rather than directly observe it.

Let $\mathcal{S}$ represent the complete, unconstrained state space of the environment state. It is feasible to express the state transition probability as $\mathcal{P}\left(s_{t+1}, \boldsymbol{a}, s_t\right) = Pr\left(s' \left| s, \boldsymbol{a}\right.\right)$, where s is current step state, s' is next step state for joint action $\boldsymbol{a}$.

5.3.3 Observation Space

As mentioned above, only the local information of the global environment is visible to each agent. At time step t, the observation of agent i is denote by:

$$O^i\left(t\right) = \left(p_i\left(t\right), w_i\left(t\right), \xi_i\left(t\right), \eta_i\left(t\right), \varpi_i\left(t\right)\right), \tag{5.4}$$

where $\eta_i\left(t\right)$ denotes channel gain, $\varpi_i\left(t\right)$ denotes sum of interference-noise power. Accordingly, both notations are expressed as:

$$\eta_i\left(t\right) = \left|h_{ii}^H\left(t\right) w_i\left(t\right) \sqrt{p_i\left(t\right)}\right|^2, \tag{5.5}$$

$$\varpi_i\left(t\right) = \sum_{j \neq i} \left|h_{ij}^H\left(t\right) w_j\left(t\right) \sqrt{p_i\left(t\right)}\right| + z_i. \tag{5.6}$$

Each D2D pair does not have a priori knowledge of the power allocation, beamforming, and achievable rates of other D2D pairs. Therefore, the D2Ds cannot take action to maximize the sum rate of the network. With a dynamic channel environment and uncertainty in channel interference, this POMDP problem can be formulated as a stochastic game in which each D2D chooses mutually independent beamforming vectors and transmit power strategies as a function of their own local observations.

5.3.4 Action Space

The action space specifies the limits or boundaries of the set of actions that an agent can choose when interacting with the environment. We consider the specific beamforming and transmission power as the action space of each agent. Unlike conventional algorithm-based power control schemes that take continuous values to meet practical circuit constraints and facilitate learning, we use discrete power levels between 0 and p_{max}. The transmit power is quantified as the L level. The set of available discrete power is given by:

$$\left\{\frac{P_{\max}}{L}, \frac{2p_{\max}}{L}, ..., P_{\max}\right\}. \tag{5.7}$$

As mentioned above, the dimension of the beamforming vector is determined by the number of transmit antennas. Thus, the size of the action space of agents with type m is $|\mathcal{A}| = A_m \times L$, and the D2D chooses the appropriate beam direction and power level and is immediately rewarded for evaluating its action.

5.3.5 Reward Function

The reward function in RL is designed based on the objective function of the optimization problem. Each agent makes a joint decision on beam direction and power level to maximize the system and rate. The learning goal of the entire multi-intelligent system is to find the optimal strategy to maximize the cumulative discount reward function, and we define the long-term reward r_i for time step t as the cumulative and discount reward as:

5.4 Multi Type Mean Field Muti Agent Reinforcement Learning

Mean Field Game has been widely employed in multi-agent environments, and its core concept is built on the aggregated information of other agents. Instead of the MFG,

which requires that all agents fulfill the homogeneous requirements, the MFTG theory is more appropriate for real-world engineering application scenarios. The agents in the MFTG scenario might be heterogeneous, and the impact of a single agent on the mean-field term and the overall effect is taken into account. This part presents the proposed problem as Multi-type Mean-field MARL (MTMFRL) formulation.

5.4.1 Multi Type Mean Field Formulation

Agents adjacent to agent i can be classified into M types. Each agent is assigned to a specific type, and each type comprises of X_m agents. We assume that the Q function (action-value function) can be additionally decomposed into subsets based on the division of agents, each subset containing agents of the same type. Let $a_m^{k_i}$ represent the action of agent k_i belonging to type m and interact with agent j. Assume that

$$Q^j\left(s, \boldsymbol{a}\right) = Q^j\left(s, a^j, a_1^{k_1}, ..., a_1^{k_{X_1}}, ..., a_M^{k_1}, ..., a_M^{k_{X_M}}\right), \tag{5.8}$$

can be factorized into the average of interactions within one type of agent, which is decomposable further pairwise. This decomposition can be seen as a generalization of the pairwise decomposition to multiple types in MFRL since each component depends on the representation of each type. Thus, the standard Q function is formulated as:

$$Q^j\left(s, \boldsymbol{a}\right) = \sum_{m=1}^{M} Q^j\left(s, a^j, a_m^{k_1}, ..., a_m^{k_{X_m}}\right) = \sum_{m=1}^{M} \frac{1}{X_m} \sum_{i=1}^{X_m} Q^j\left(s, a^j, a_m^{k_i}\right). \tag{5.9}$$

5.4.2 Multi Type Mean Field Approximation

We employ the one-hot representations in Yang et al. for actions, the action of agent k belonging to type m can be represented as $a_m^k = \overline{a}_m^j + \delta^{j,k}$, where $\overline{a}_m^j$ denotes the mean action of adjacent agents of j and $\delta^{j,k}$ denotes the difference between an agent's action and the mean action of its subsets. Let $\overline{a}_m^j = \frac{1}{X_m} \sum_{i=1}^{X_m} a_m^{k_i}$, the mean-field theorem enables the approximation of the Q function by the Taylor approximation as:

$$Q^j\left(s, \boldsymbol{a}\right) \approx Q^j\left(s, \overline{a}_1^j, \overline{a}_2^j, ..., \overline{a}_M^j\right). \tag{5.10}$$

This hierarchical mean-field emphasizes interactions between types because agents of the same type have identical properties, and their interactions can be estimated using a standard mean field. We modified the mean field Q function to include a finite number of types, each with a corresponding mean-field, in light of the fact that agents

of distinct types exhibit greater variation. This approximation reduces the intricacy of agent interactions and the variance of the Q function while implicitly maintaining the global interactions between any two agents.

5.4.3 Multi Type Mean Field Update

At time slot t, the mean field Q function for agent j can be updated iteratively:

$$Q_{t+1}^{j}\left(s,\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right)=\left(1-\alpha\right)Q_{t}^{j}\left(s,\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right)+\alpha\left[r^{j}+\gamma V_{t}^{j}\left(s'\right)\right], \tag{5.11}$$

where α denotes the learning rate. The mean-field value function of agent j at time step t is estimated as:

$$V_{t}^{j}\left(s'\right)=\sum_{a^{j}}\pi_{t}^{j}\left(a^{j}\left|s,\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right.\right)\mathbb{E}_{a^{-j}\sim\pi_{t}^{-j}}\left[Q_{t}^{j}\left(s',a^{j},\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right)\right], \tag{5.12}$$

where π_t^j represents the action policy of agent j. The Boltzmann policy chooses the strategy for each agent j:

$$\pi_{t}^{j}\left(a^{j}\left|s,\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right.\right)=\frac{\exp\left(\varphi Q_{t}^{j}\left(s,a^{j},\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right)\right)}{\sum_{a^{j'}\in A^{j}}\exp\left(\varphi Q_{t}^{j}\left(s,a^{j},\overline{a}_{1}^{j},...,\overline{a}_{M}^{j}\right)\right)}, \tag{5.13}$$

where φ is the Boltzmann softmax hyper-parameter. Through iterative equations, all agents' mean actions and corresponding strategies are continually updated. It will eventually converge to a fixed point close to the Nash equilibrium. The convergence was proved in.

5.4.4 Multi Type Mean Field Solution

A DDQN-based algorithm is proposed to estimate the Multi Type Mean Field (MTMF) Q function. The framework is illustrated in Figure 5.2, DDQN employs two neural networks to guarantee network performance stability . The target Q network with parameter θ^- has the same network structure as the training Q network with parameter θ. θ is updated at each step, but θ^- is updated to the value of θ at each of several steps, after which the parameter values are fixed for the remaining steps.

Experience replay enhances learning efficiency by reusing training data and minimizing serial data correlation . At each time step t, after the transition experience data

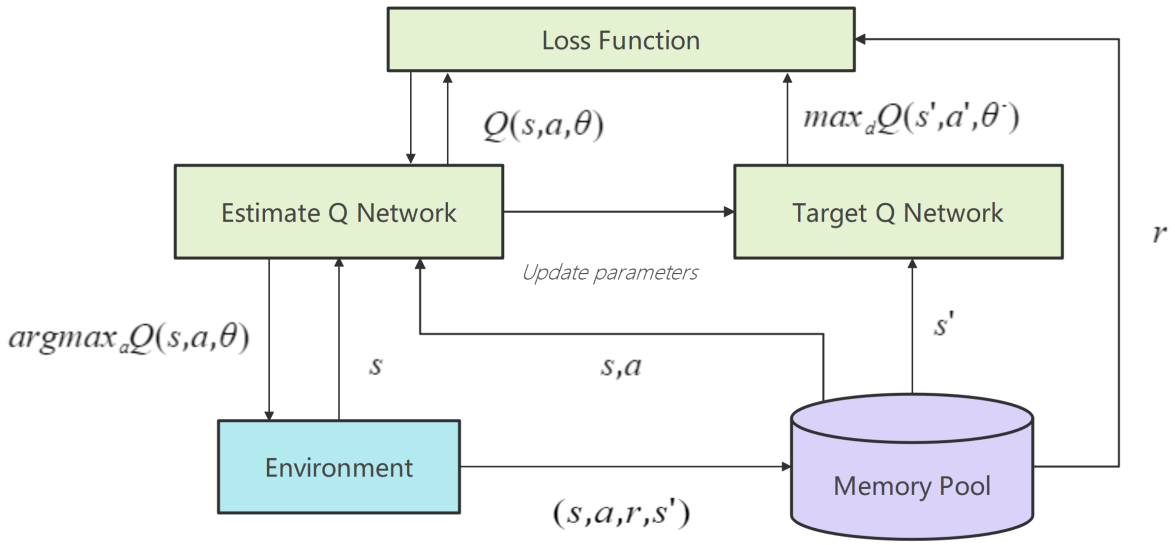

Figure 5.2: Illustration of the multi-agent DDQN framework.

$\boldsymbol{e}_t = (s_t, \boldsymbol{a}_t, \boldsymbol{r}_{t+1}, s_{t+1})$ is collected by agent i into its replay memory, each D2D agent uniformly samples a mini-batch of experiences e from the memory pool and updates the weights of its Q network using stochastic gradient descent. The loss function is formulated as follows:

$$\mathcal{L}_{\theta^j} = \left(y^j - Q^j\left(s, \boldsymbol{a}, \theta\right)\right)^2, \tag{5.14}$$

where $\boldsymbol{a} = \left[a^j, \overline{a}_1^j, ..., \overline{a}_M^j\right]$ denotes the mean field joint action, and

$$y_t^i = r_{t+1}^j + \gamma Q^j\left(s^{t+1}, arg\max_{\boldsymbol{a}'} Q^j\left(s^{t+1}, \boldsymbol{a}', \theta\right), \theta^-\right), \tag{5.15}$$

denotes the mean field value calculated by the update of the target network. It is used to calculate the Temporal Difference (T.D.) error. Thus, the gradient of the loss function is calculated as:

$$\nabla_{\theta^j}\mathcal{L}_{\theta^j} = 2\left(y^j - Q^j\left(s, \boldsymbol{a}, \theta\right)\right) \times \nabla_{\theta^j} Q^j\left(s, \boldsymbol{a}, \theta\right). \tag{5.16}$$

Algorithm 5 describes the step of MTMF-Q algorithm.

5.5 Simulation

To analyze and evaluate the performance of the proposed algorithm, we compare it with the MF-Q, DDQN, and random strategy under the same environment.

Algorithm 5 Multi Type Mean Field Double Deep Q Network.

Initialize replay memory D.
Initialize target Q and estimate Q network with parameters θ^- and θ.
for Episode ={$1, ..., E$} do **do**
  for Step = {$1, ..., T$} do **do**
    Select action a_t^j according to Q-value with greedy ε for each agent.
    Caculate new mean action $\overline{a}_1^j, ..., \overline{a}_M^j$ for each type.
    Excute joint action and observe reward $\boldsymbol{r_t^j}$ and next state s_{t+1}^j.
    Store tuple $\langle s_t, \boldsymbol{a}, r, s_{t+1} \rangle$ in replay buffer.
    Sample random mini-batch from D.
    Perform gradient decent on loss function wrt θ in estimate Q.
    Set y_t^i
    Update $\theta^- = \theta$ every C steps
  end for
end for

5.5.1 Environment Setting

In this section, we simulate a single-cell scenario for a heterogeneous D2D wireless network, considering two types of D2D links: 1) SISO links and 2) 2×1 MISO links randomly distributed in a circular area with a radius of 500m. The maximum transmit power P_{max} and noise power are set to 30 dBm and -174 dBm, respectively. The communication distance of each link is limited between d_{min} = 2m and d_{max} = 50m. Large-scale fading is modeled as $\beta = -120.9 - 37.6 \log_{10}(d) + 10 \log_{10}(z)$ through telecommunications long-term evolution (LTE) standard [77]. Small-scale fading is represented by a complex Gaussian variable with zero mean and one variance. Using the distance-dependent scaling model in [73], the path loss and shadowing caused by large-scale fading are calculated. In experiments evaluating the performance of the proposed method under different system parameters, the ratio of SISO and MISO links is set to 1:1, while the network topology for each D2D sample is determined independently. Hyperparameters of DDQN are summarized in Table 5.1

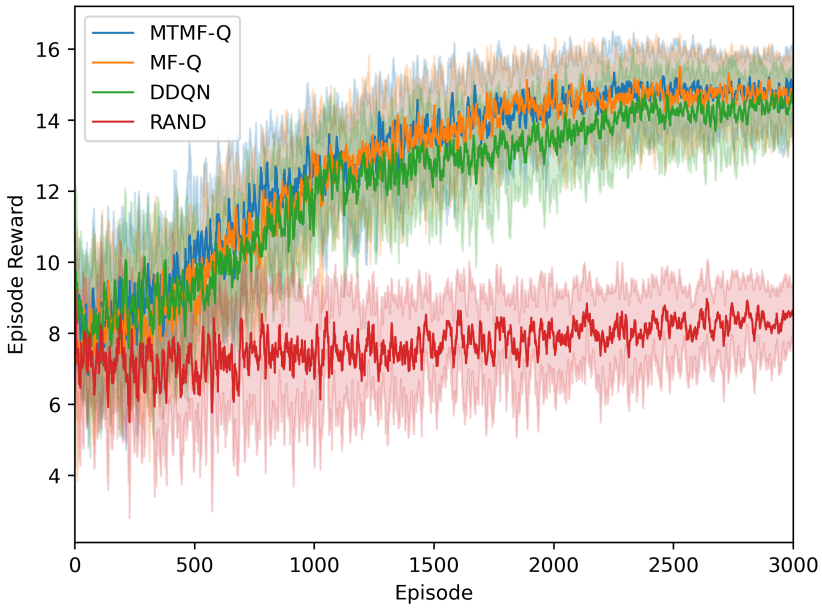

Figure 5.3: Training performances with 8 agents.

Table 5.1: DDQN hyperparameters

Parameter	Value
Number of Episode	3000
Time step per Episode	50
Size of the mini-batch	32
Size of the memory step	500
Learning rate	0.001
Learning rate decay factor	0.0001
Discount factor	0.99
Hidden layers	5
Activation function	ReLU

5.5.2 Performance Comparison

We conducted a study deploying various numbers of D2D links to evaluate the performance of the proposed MTMF-Q algorithm in heterogeneous networks. Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6 illustrates the episode rewards obtained during the training phase, with the random strategy considered as a lower bound for comparison purposes. It is important to note that in our experiments, none of the algorithms considered credit distribution or inter-agent communication. Each agent solely relied on local observations to update its actions. We realize each algorithm with various initial

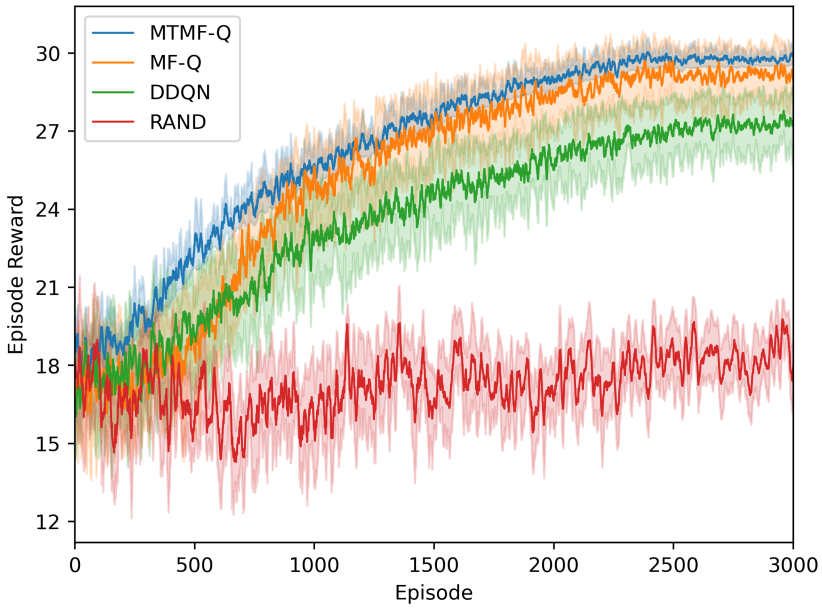

Figure 5.4: Training performances with 16 agents.

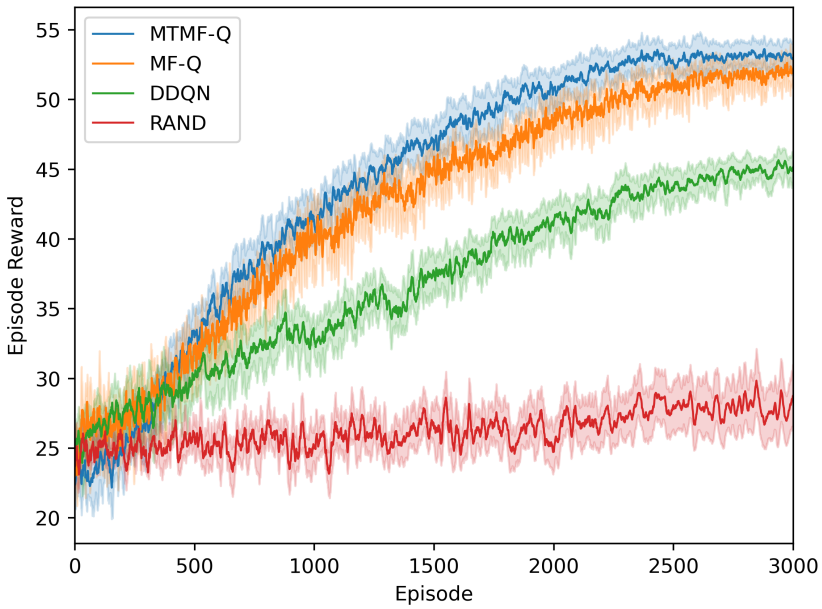

Figure 5.5: Training performances with 32 agents.

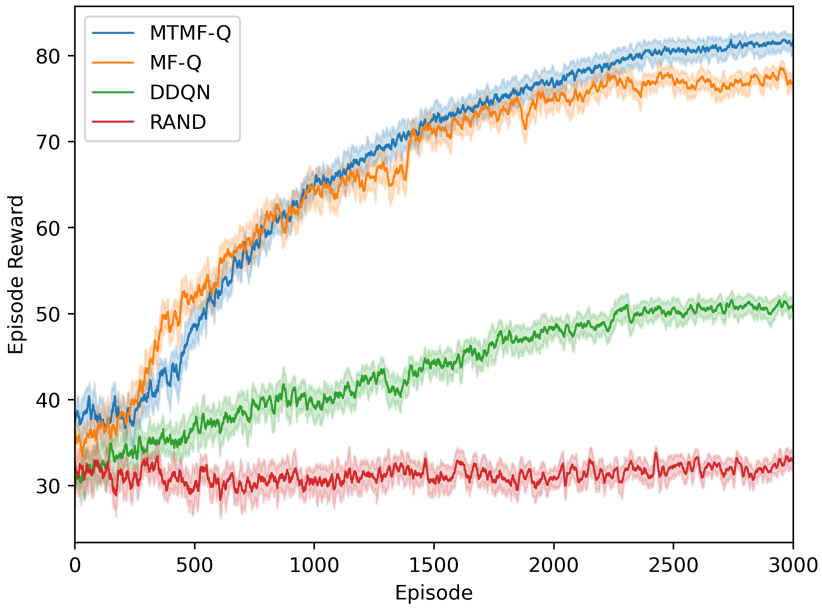

Figure 5.6: Training performances with 64 agents.

random seeds and smooth the curves in each plot to enhance readability.

When the network size is small, the performance of different classes of RL algorithms is comparable. However, as the network size grows, the DDQN algorithm proves to be inferior to the MF-based approach and faces challenges in convergence. This can be attributed to the limited observation information available to each agent, which hampers the effectiveness of individual agent updates. On the other hand, the MF-based algorithm leverages the mean field of the two agent classes (MISO, SISO) as the decision-making factor. The Q values used for action selection are influenced by internal changes in the action parameters. Notably, the MTMF-Q algorithm outperforms the standard MFQ algorithm, and the performance gap between the two widens as the number of D2Ds increases. The MTMF-Q approach excels in handling the multi-type mean field of heterogeneous agents with distinct action spaces, benefiting from its foundation in MFTG theory. The efficiency of the proposed algorithm is demonstrated by the stable and rapid increase in rewards observed in large-scale agent networks.

Figure 5.7 provides insights into the performance of the D2D network in relation to different power levels. We varied the power level of the D2D network, while keeping other parameters fixed (D2D=32), to assess the effectiveness of the proposed method in a large action space. When the action space is small, there is no significant difference in performance between the MF-based approach and DDQN. However, as the power level becomes more finely divided, resulting in increased action space, the MF-based

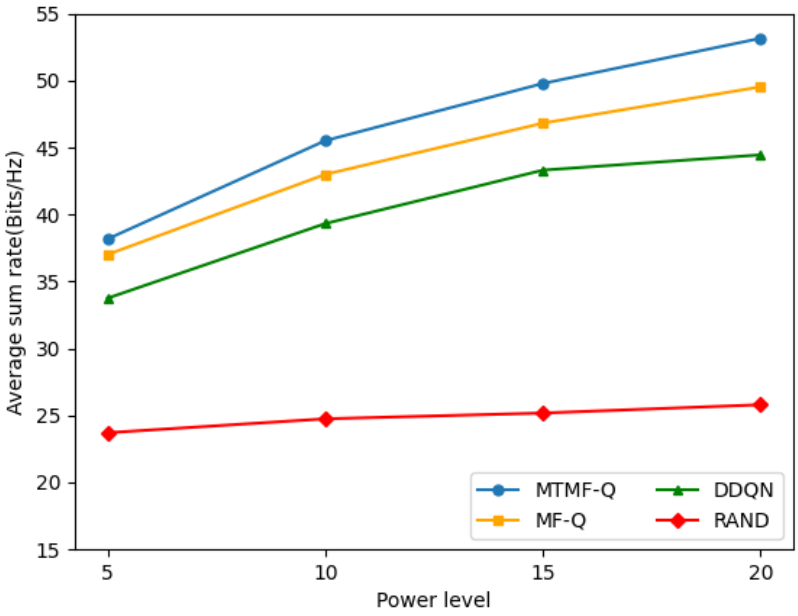

Figure 5.7: Network's sum rate versus power level

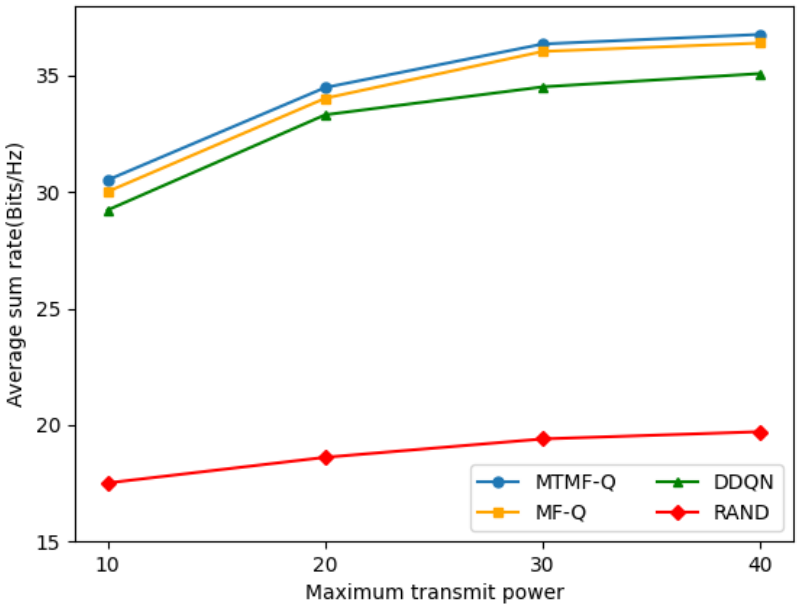

Figure 5.8: Network's sum rate versus maximum transmit power.

method achieves higher average rewards. Notably, the MTMF-Q algorithm continues to outperform the other algorithms in terms of improving the network's sum rate.

Figure 5.8 depicts the sum rate of the D2D network as a function of different maximum transmit powers. The curve exhibits a slow rise within the range of 10 dBm to 30 dBm and stabilizes within the range of 30 dBm to 40 dBm. In the range of 10 dBm to 30 dBm, the gradual growth of the curve can be attributed to the limited total achievable rate resulting from the low available transmit power. As the transmit power increases within the range of 30 dBm to 40 dBm, although higher transmission power becomes available for the D2D links, the interference from other devices, incorporated in the reward function of the sum rate, also increases. Consequently, the marginal benefit of power on the sum rate improvement diminishes significantly. Overall, the MTMF algorithm exhibits effectiveness and stability across different network parameters.

We propose a novel graph neural network called Heterogeneous Edge Feature Enhanced Graph Attention Networks (HEGAT) to address the above challenges. HEGAT considers both graph heterogeneity as well as edge features. It combines node and edge features based on a two-layer attention mechanism, including node-level and edge-level attention. Specifically, we want to know how vital node-based and edge-based neighbors are and how each node fits into the node-level attention layer. Similarly, the embedding of each edge is generated in the edge-level attention layer.

5.6 Conclusion to Chapter 5

This chapter focuses on the beamforming and power control problems in heterogeneous D2D networks with multiple link types, with the objective of maximizing the sum rate of the network. As the number and types of D2D devices increase, calculating rewards individually for each device leads to significant performance degradation, which is not desirable when applying existing MFRL methods. To tackle this challenge, we propose an enhanced MFRL approach called MTMF-Q, which integrates MFTG theory with MARL in heterogeneous networks to achieve optimal joint beamforming and power allocation. The MTMF-Q algorithm is designed to address the limitations of traditional MARL techniques in scenarios with a large number of agents. By considering the interactions between each agent and different types of mean fields, the algorithm achieves efficient resource allocation and overcomes the scalability issues of individual reward calculations. Through experimental evaluations, we demonstrate that MTMF-Q

outperforms the state-of-the-art MF-Q method in large-scale scenarios. Moreover, the proposed algorithm exhibits stable performance across networks with different parameters, enhancing its effectiveness in practical deployments.

In conclusion, our proposed approach has the potential to overcome the limitations of traditional MARL techniques, allowing for the deployment of ultra-dense D2D communication networks. In future research, we aim to explore communication networks with a greater variety of device types, moving closer to realistic scenarios. Additionally, we plan to investigate the application of MF-MARL in allocating other communication resources, thus further enhancing the practical value and applicability of the algorithm.

Conclusion and Future Works

This chapter serves as the concluding section of the dissertation, summarizing the research results and contributions made during my PhD, as well as outlining potential future research directions.

5.7 Conclusion

The research primarily focuses on two key techniques in wireless networks: 1) heuristic algorithms and 2) machine learning. The paper is organized around the theme of resource allocation, and the conclusions of each chapter are presented below. The main results of the work are as follows:

1. Chapter 1 introduces the fundamentals and methods of resource allocation in cellular networks. The problem addressed in this paper lacks closed-form expressions for optimization variables, making heuristic algorithms a suitable approach. Simulation experiments demonstrate the advantages of metaheuristic algorithms in resource allocation and interference management. However, they often require numerous stochastic searches to attain satisfactory solutions, which may not meet the real-time decision-making demands of communication networks.

2. In Chapter 2, the beamforming design problem in more complex D2D networks is investigated. A PSO-based supervised learning framework is developed to tackle continuous optimization issues in wireless networks. This framework leverages the robust learning capabilities of residual networks and the rapid processing speeds of deep learning. Additionally, a workflow for addressing general resource allocation problems is presented, involving the generation of labeled datasets through optimization techniques and their offline training using machine learning methods. Deep neural networks exhibit significant potential for shifting time-consuming computations to offline training and solving complex, hard-to-model problems.

Nevertheless, they necessitate substantial datasets for improved learning of resource allocation policy mappings.

3. Chapter 3 explores a supervised learning framework based on graph neural networks, building upon the workflow introduced in Chapter 2. The offline-trained DNN models are replaced by MPGNN and GAT, effectively harnessing the information embedded in the communication network's topology. Experimental results reveal that suboptimal model performance can be achieved with only a few training samples.

4. In Chapter 4, the resource allocation problem in more intricate, heterogeneous D2D wireless networks is investigated. An unsupervised learning algorithm named HEGAT is developed to jointly optimize beamforming and power allocation strategies. This model considers different types of communication networks and edge information while using an attention mechanism to determine the most critical edges and nodes. The deep learning approach presented here is versatile and surpasses the performance of the best graph learning-based resource optimization methods.

5. Chapter 5 delves into the resource allocation problem in D2D wireless networks with large-scale heterogeneity. The machine learning-based methods proposed in the previous chapters concentrate decision-making, which is often infeasible for real-world applications in large-scale networks. A combination of mean-field games and multi-agent reinforcement learning is employed to achieve decentralized decision-making for D2D resource allocation with limited CSI. This approach demonstrates the potential to operate in large networks and handle partially missing input features.

These machine-learning approaches offer significant enhancements to radio resource allocation, including real-time execution, scalability, generalization, and decentralized implementation.

5.8 Future Works

Based on the work presented in this paper, several promising future research directions are worth exploring, which are summarized below:

5.8.1 Further Extension of GNN Approach

The proposed GNN method is currently implemented in a centralized manner for solving the wireless resource allocation problem. However, centralized approaches may raise privacy concerns. The simulation results have demonstrated the GNN's excellent capabilities, including scalability to large-scale networks, generalization to different system settings without retraining, and robustness to corrupted input features. To preserve these desirable features and address privacy concerns, exploring a distributed deployment of GNNs is essential. Such an approach is more suitable for wireless communication applications than centralized deployment. One potential direction is to combine GNN with MARL to achieve this goal.

5.8.2 Further Extension of RL Approach

The RL approach proposed in this paper serves as a preliminary step toward realizing resource allocation for large-scale D2D networks. Several aspects can be further investigated, as outlined below:

1. The proposed RL method currently operates in a supervised manner, relying on labeled datasets for training. The experience playback stores data in the form of tuples in the experience pool but does not fully utilize the topological information of the communication network, resulting in suboptimal learning efficiency. Investigating how to effectively leverage the topological information of communication networks is a promising research direction.

2. The scalability of the RL method can be enhanced. The RL model in this paper only considers a relatively smaller action state space, involving discretization processing to train the model and jointly optimize beamforming and power allocation strategies. It would be valuable to explore a decentralized RL approach capable of handling a broader range of network resources, such as memory, link scheduling, and latency. Additionally, investigating large-scale action state spaces is another promising research direction.

Bibliography

[1] Fooladivanda D., Rosenberg C. Joint resource allocation and user association for heterogeneous wireless cellular networks // IEEE Transactions on Wireless Communications. 2012. Vol. 12. N. 1. P. 248–257.

[2] Shen K., Yu W. Fractional programming for communication systems Part I: Power control and beamforming // IEEE Transactions on Signal Processing. 2018. Vol. 66. N. 10. P. 2616–2630.

[3] Shi Q., Razaviyayn M., Luo Z., He C. An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel // IEEE Transactions on Signal Processing. 2011. Vol. 59. N. 9. P. 4331-4340.

[4] Zhang X., Nakhai M. R., Zheng, G., Lambotharan S., Ottersten B. Calibrated learning for online distributed power allocation in small-cell networks // IEEE Transactions on Signal Processing. 2019. Vol. 67. N. 11. P. 8124–8136.

[5] Sun H., Chen X., Shi Q., Hong M., Fu X., Sidiropoulos N. D. Learning to optimize: Training deep neural networks for interference management // IEEE Transactions on Signal Processing. 2018. Vol. 66. N. 20. P. 5438-5453.

[6] Eisen M., Zhang C., Chamon L. F., Lee D. D., Ribeiro A. Learning optimal resource allocations in wireless systems // IEEE Transactions on Signal Processing. 2019. Vol. 67. N. 10. P. 2775-2790.

[7] Sun Q., Wu H., Petrosian, O. Optimal Power Allocation Based on Metaheuristic Algorithms in Wireless Network // Mathematics. 2022. Vol. 10. N. 18. P. 3336.

[8] Kumar M., Sharma, S. C. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing // Neural Computing and Applications. 2020. Vol. 32. P. 12103–12126.

[9] Khanolkar S., Sharma N., Anpalagan A. Energy-Efficient Resource Allocation in Underlay D2D Communication using ABC Algorithm // Wireless Personal Communications. 2022. Vol. 12. N. 5. P. 1443–1468.

[10] Sun H., Chen X., Shi Q., Hong M., Fu X., Sidiropoulos N. D. Learning to optimize: Training deep neural networks for interference management // IEEE Transactions on Signal Processing. 2018. Vol. 66. N. 20. P. 5438–5453.

[11] Blough D. M., Resta G., Santi P. Approximation algorithms for wireless link scheduling with SINR-based interference // IEEE/ACM Transactions on networking. 2020. Vol. 18. N. 20. P. 5438–5453.

[12] Chowdhury A., Verma G., Rao C., Swami A., Segarra S. Unfolding WMMSE using graph neural networks for efficient power allocation // IEEE Transactions on Wireless Communications. 2021. Vol. 20. N. 9. P. 6004–6017.

[13] Zhang X., Zhao H., Xiong J., Liu X., Zhou L., Wei J. Scalable power control/beamforming in heterogeneous wireless networks with graph neural networks // IEEE Global Communications Conference (GLOBECOM). 2021. P. 01–06.

[14] Zhang X., Zhao H., Xiong J., Liu X., Zhou L., Wei J. Scalable power control/beamforming in heterogeneous wireless networks with graph neural networks // Innovations in multi-agent systems and applications-1. 2010. P. 183–221.

[15] Li T., Zhu K., Luong N. C., Niyato D., Wu Q., Zhang Y., Chen B. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey // IEEE Communications Surveys and Tutorials. 2021. Vol. 24. N. 2. P. 1240–1279.

[16] Rouwet W. Open Radio Access Network (O-RAN) Systems Architecture and Design // Elsevier. 2022

[17] Li Y., Han S., Yang C. Multicell power control under rate constraints with deep learning // IEEE Transactions on Wireless Communications. 2021. Vol. 20. N. 12. P. 7813–7825.

[18] Marini F., Walczak B. Particle swarm optimization (PSO). A tutorial. // Chemometrics and Intelligent Laboratory Systems. 2015. Vol. 149. P. 153–165.

[19] Li J. Y., Zhan Z. H., Liu R. D., Wang C., Kwong S., Zhang J. Generation-level parallelism for evolutionary computation: A pipeline-based parallel particle swarm optimization // IEEE Transactions on Cybernetics. 2020. Vol. 51. N. 10. P. 4848–4859.

[20] Schluter M., Egea J. A., Banga J. R. Extended ant colony optimization for non-convex mixed integer nonlinear programming // Computers and Operations Research. 2008. Vol. 36. N. 7. P. 2217–2229.

[21] Price K. V. Differential evolution. In Handbook of optimization: From classical to modern approach // Handbook of optimization: From classical to modern approach. 2013. P. 187–214.

[22] Elsayed S. M., Sarker R. A., Essam D. L. Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems // IEEE Congress of Evolutionary Computation (CEC). 2011. P. 1041–1048.

[23] Brest J., Greiner S., Boskovic B., Mernik M., Zumer V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems // IEEE transactions on evolutionary computation. 2006. Vol. 10. N. 6. P. 646–657.

[24] Corana A., Marchesi M., Martini C., Ridella S. Minimizing multimodal functions of continuous variables with the “simulated annealing” algorithm—Corrigenda for this article is available here // ACM Transactions on Mathematical Software. 1987. Vol. 13. N. 3. P. 262–280.

[25] Wales D. J., Doye J. P. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms // The Journal of Physical Chemistry A. 1997. Vol. 101. N. 28. P. 5111–5116.

[26] Hansen N. The CMA evolution strategy: a comparing review // Towards a new evolutionary computation: Advances in the estimation of distribution algorithms. 2006. P. 75–102.

[27] Ansari R. I., Chrysostomou C., Hassan S. A., Guizani M., Mumtaz S., Rodriguez J., Rodrigues J. J. 5G D2D networks: Techniques, challenges, and future prospects // IEEE Systems Journal. 2017. Vol. 12. N. 4. P. 3970–3984.

[28] Li L., Xu Y., Yin J., Liang W., Li X., Chen W., Han Z. Deep reinforcement learning approaches for content caching in cache-enabled D2D networks // IEEE Internet of Things Journal. 2019. Vol. 7. N. 1. P. 544–557.

[29] Ruan L., Dias M. P. I., Wong E. Deep neural network supervised bandwidth allocation decisions for low-latency heterogeneous e-health networks // Journal of Lightwave Technology. 2019. Vol. 37. N. 16. P. 4147–4154.

[30] Huang H., Liu M., Gui G., Gacanin H., Sari H., Adachi F. Unsupervised learning-inspired power control methods for energy-efficient wireless networks over fading channels // IEEE Transactions on Wireless Communications. 2022. Vol. 21. N. 11. P. 9892–9905.

[31] Jiang Y., Liu Q., Zheng F., Gao X., You X. Energy-efficient joint resource allocation and power control for D2D communications // IEEE Transactions on Vehicular Technology. 2015. Vol. 65. N. 8. P. 6119–6127.

[32] Sun H., Chen X., Shi Q., Hong M., Fu X., Sidiropoulos N. D. Learning to optimize: Training deep neural networks for interference management // IEEE Transactions on Signal Processing. 2015. Vol. 66. N. 20. P. 5438–5453.

[33] Su L., Ji Y., Wang P., Liu F. Learning to optimize: Training deep neural networks for interference management // IEEE wireless communications and networking conference (WCNC). 2013. P. 129–133.

[34] Lu L., He D., Li G. Y., Yu X. Graph-based robust resource allocation for cognitive radio networks // IEEE Transactions on Signal Processing. 2015. Vol. 63. N. 14. P. 3825–3836.

[35] Xu M. Understanding graph embedding methods and their applications // SIAM Review. 2021. Vol. 63. N. 4. P. 825–853.

[36] Zhang S., Tong H., Xu J. Graph convolutional networks: a comprehensive review // Computational Social Networks. 2019. Vol. 6. N. 1. P. 1–23.

[37] Velickovic P., Cucurull G., Casanova A., Romero A., Lio P., Bengio Y. Graph attention networks // stat. 2017. Vol. 1050. N. 20. P. 10–48550.

[38] Wu Z., Pan S., Chen F., Long G., Zhang C., Philip S. Y. A comprehensive survey on graph neural networks // IEEE transactions on neural networks and learning systems. 2020. Vol. 32. N. 1. P. 4–24.

[39] Gilmer J., Schoenholz S. S., Riley P. F., Vinyals O., Dahl G. E. Neural message passing for quantum chemistry // International conference on machine learning. 2017. P. 1263–1272.

[40] Shen Y., Shi Y., Zhang J., Letaief K. B. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis // IEEE Journal on Selected Areas in Communications. 2020. Vol. 39. N. 1. P. 101–115.

[41] Eisen M., Ribeiro A. Optimal wireless resource allocation with random edge graph neural networks // IEEE Transactions on signal Processing. 2020. Vol. 68. P. 2977–2991.

[42] Chowdhury A., Verma G., Rao C., Swami A., Segarra S. Efficient power allocation using graph neural networks and deep algorithm unfolding // IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2021 P. 4725–4729.

[43] Schlichtkrull M., Kipf T. N., Bloem P., Van Den Berg R., Modeling relational data with graph convolutional networks // Semantic Web: 15th International Conference. 2018. P. 593–607.

[44] Gong L., Cheng Q. Exploiting edge features for graph neural networks // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. P. 9211–9219.

[45] Guo J., Yang C. Learning power control for cellular systems with heterogeneous graph neural network // IEEE Wireless Communications and Networking Conference (WCNC). 2021. P. 1–6.

[46] Feng D., Lu L., Yuan-Wu Y., Li G. Y., Li S., Feng G. Device-to-device communications in cellular networks // IEEE Communications Magazine. 2014. Vol. 52. N. 4. P. 49–55.

[47] Zhang L., Xiao M., Wu G., Alam M., Liang Y. C., Li S. A survey of advanced techniques for spectrum sharing in 5G networks // IEEE Wireless Communications. 2017. Vol. 24. N. 5. P. 44–51.

[48] Ye P. G., Liang W., Lu Q., Xiao R. F., Guo Z. Y., Sun K. X. Spiking mean field multi-agent reinforcement learning for dynamic resources allocation in D2D networks // International Conference on Ubiquitous Computing and Communications. 2021. P. 60–67.

[49] Xu Y., Gui G., Gacanin H., Adachi F. A Survey on Resource Allocation for 5G Heterogeneous Networks: Current Research, Future Trends and Challenges // IEEE Communications Surveys and Tutorials. 2021. Vol. 23. N. 2. P. 668–695.

[50] Jiang Y., Zou Y., Guo H., Tsiftsis T. A., Bhatnagar M. R., de Lamare R. C., Yao Y. D. Joint power and bandwidth allocation for energy-efficient heterogeneous cellular networks // IEEE Transactions on Communications. 2021. Vol. 67. N. 9. P. 6168–6178.

[51] Abrardo A., Moretti M. Distributed power allocation for D2D communications underlaying/overlaying OFDMA cellular networks // IEEE Transactions on Wireless Communications. 2016. Vol. 16. N. 3. P. 1466–1475.

[52] Yu C. H., Doppler K., Ribeiro C. B., Tirkkonen O. Resource sharing scheme for device-to-device communication underlaying cellular networks // IEEE transactions on communications. 2015. Vol. 63. N. 12. P. 4838–4848.

[53] Li T., Zhu K., Luong N. C., Niyato D., Wu Q., Zhang Y., Chen B. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey // IEEE Communications Surveys and Tutorials. 2022. Vol. 24. N. 2. P. 1240–1279.

[54] Barik P. K., Shukla A., Datta R., Singhal C. A resource sharing scheme for intercell D2D communication in cellular networks: A repeated game theoretic approach // IEEE Transactions on Vehicular Technology. 2020. Vol. 69. N. 7. P. 7806–7820.

[55] Lee W., Lee K. Resource allocation scheme for guarantee of QoS in D2D communications using deep neural network // IEEE Communications Letters. 2020. Vol. 25. N. 3. P. 887–891.

[56] Li L., Xu Y., Yin J., Liang W., Li X., Chen W., Han Z. Deep reinforcement learning approaches for content caching in cache-enabled D2D networks // IEEE Internet of Things Journal. 2019. Vol. 7. N. 1. P. 554–557.

[57] Yan Y., Zhang B., Li C., Su C. Cooperative caching and fetching in d2d communications-a fully decentralized multi-agent reinforcement learning approach // IEEE Transactions on Vehicular Technology. 2019. Vol. 69. N. 12. P. 16095–16109.

[58] Yan Y., Zhang B., Li C., Su C. Cooperative caching and fetching in d2d communications-a fully decentralized multi-agent reinforcement learning approach // IEEE Transactions on Vehicular Technology. 2019. Vol. 69. N. 12. P. 16095–16109.

[59] Tang Q., Xie R., Yu F. R., Huang T., Liu Y. Decentralized computation offloading in IoT fog computing system with energy harvesting: A dec-POMDP approach // IEEE Internet of Things Journal. 2020. Vol. 7. N. 6. P. 4898–4911.

[60] Guo D., Tang L., Zhang X., Liang Y. C. Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning // IEEE Transactions on Vehicular Technology. 2020. Vol. 69. N. 11. P. 13124–13138.

[61] Mseddi A., Jaafar W., Moussaid A., Elbiaze H., Ajib W. Collaborative D2D pairing in cache-enabled underlay cellular networks // IEEE Global Communications Conference (GLOBECOM). 2021. P. 1–6.

[62] Mseddi A., Jaafar W., Moussaid A., Elbiaze H., Ajib W. Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning // IEEE Transactions on Vehicular Technology. 2020. Vol. 69. N. 11. P. 13124–13138.

[63] Rashid T., Samvelyan M., De Witt C. S., Farquhar G., Foerster J., Whiteson S. Monotonic value function factorisation for deep multi-agent reinforcement learning // The Journal of Machine Learning Research. 2020. Vol. 21. N. 1. P. 7234–7284.

[64] Li L., Cheng Q., Tang X., Bai T., Chen W., Ding Z., Han Z. Resource allocation for NOMA-MEC systems in ultra-dense networks: A learning aided mean-field

game approach // IEEE Transactions on Wireless Communications. 2020. Vol. 20. N. 3. P. 1487–1500.

[65] Yang Y., Luo R., Li M., Zhou M., Zhang W., Wang J. Resource allocation for NOMA-MEC systems in ultra-dense networks: A learning aided mean-field game approach // International conference on machine learning. 2018. P. 5571–5580.

[66] Yang C., Li J., Semasinghe P., Hossain E., Perlaza S. M., Han Z. Distributed interference and energy-aware power control for ultra-dense D2D networks: A mean field game // IEEE Transactions on Wireless Communications. 2016. Vol. 16. N. 2. P. 1205–1217.

[67] Chen D., Qi Q., Zhuang Z., Wang J., Liao J., Han Z. Mean field deep reinforcement learning for fair and efficient UAV control // IEEE Internet of Things Journal. 2016. Vol. 8. N. 2. P. 813–828.

[68] Zhang H., Tang H., Hu Y., Wei X., Wu C., Ding W., Zhang X. P. Heterogeneous Mean-Field Multi-Agent Reinforcement Learning for Communication Routing Selection in SAGI-Net // IEEE 96th Vehicular Technology Conference. P. 1–5.

[69] Lixin L., Yan S., Cheng Q., Dawei W, Wensheng L, Wei C. Optimal trajectory and downlink power control for multi-type UAV aerial base stations // Chinese Journal of Aeronautics. 2021. Vol. 34. N. 9. P. 11–23.

[70] Shen Y., Shi Y., Zhang J., Letaief K. B. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis // IEEE Journal on Selected Areas in Communications. 2020. Vol. 39. N. 1. P. 101–115.

[71] Subramanian S. G., Poupart P., Taylor M. E., Hegde N. Multi type mean field reinforcement learning // arXiv preprint arXiv:2002.02513. 2020.

[72] Guo X., Hu A., Xu R., Zhang J. Learning mean-field games // Advances in Neural Information Processing Systems. 2019. Vol. 32.

[73] Shen Y., Shi Y., Zhang J., Letaief K. B. Graph neural networks for scalable radio resource management: Architecture design and theoretical analysis // IEEE Journal on Selected Areas in Communications. 2020. Vol. 39. N. 1. P. 101–115.

[74] Djehiche B., Tcheukam A., Tembine H. Mean-field-type games in engineering // arXiv preprint arXiv:1605.03281. 2016.

[75] Chen T., Zhang X., You M., Zheng G., Lambotharan S. Chen T, Zhang X, You M, et al. A GNN-based supervised learning framework for resource allocation in wireless IoT networks // IEEE Internet of Things Journal. 2021. Vol. 9. N. 3. P. 1712–1724.

[76] Alsharif M. H., Kelechi A. H., Albreem M. A., Chaudhry S. A., Zia M. S., Kim S. Sixth generation (6G) wireless networks: Vision, research activities, challenges and potential solutions // Symmetry. 2020. Vol. 12. N. 4. P. 676.

[77] Frias Z., Perez J. Techno-economic analysis of femtocell deployment in long-term evolution networks // EURASIP journal on wireless communications and networking. 2012. P. 1–15.

[78] Zhang K., Yang Z., Başar T. Multi-agent reinforcement learning: A selective overview of theories and algorithms // Handbook of reinforcement learning and control. 2021. P. 321–384.

[79] Van Hasselt H., Guez A., Silver D. Deep reinforcement learning with double q-learning // Proceedings of the AAAI conference on artificial intelligence. 2021. Vol. 30. N. 1.

[80] Chen T., Zhang X., You M., Zheng G., Lambotharan S. Chen T, Zhang X, You M, et al. A deeper look at experience replay // arXiv preprint arXiv:1712.01275. 2017.

[81] Guo H., Liang Y. C., Chen J., Larsson E. G. Weighted sum-rate maximization for intelligent reflecting surface enhanced wireless networks // 2019 IEEE Global Communications Conference (GLOBECOM). 2019. P. 1–6.

[82] Zhao J., Liu Y., Chai K. K., Chen Y., Elkashlan M., Alonso-Zarate J. A NOMA-based D2D communications: Towards 5G // IEEE global communications conference (GLOBECOM). 2021. Vol. 9. N. 3. P. 1712–1724.

[83] LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. 2015. Vol. 521. N. 7553. P. 436–444.

[84] Sun Q and Zhao C and Petrosian O and Li Y. Power allocation in wireless cellular networks: stochastic algorithm-based approach // Management processes and sustainability. 2022. Vol. 9. N. 1. P. 357–362.

[85] Sun Q., Zhang Y., Wu H., Petrosian O. Resource Allocation in Heterogeneous Network with Supervised GNNs // International Conference on Swarm Intelligence. 2023. P. 350–361.

[86] Chowdhury A., Verma G., Rao C., Swami A., Segarra S. Unfolding WMMSE using graph neural networks for efficient power allocation // IEEE Transactions on Wireless Communications. 2021. Vol. 20. N. 9. P. 6004–6017.

[87] Qiushi S., Yang H., Petrosian O. Graph attention network enhanced power allocation for wireless cellular system // Informatics and Automation. 2024. Vol. 23. N. 1. P. 259–283.

[88] Sun Q., Zhang Y., Wu H., Petrosian O. L. Deep Neural Network based Resource Allocation in D2D Wireless Networks // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2023. Vol. 19. N. 4. P. 411–421.

[89] Wang X., Ji H., Shi C., Wang B., Ye Y., Cui P., Yu P. S. Heterogeneous graph attention network // The world wide web conference. 2019. P. 2022–2032.

[90] Velickovic P., Cucurull G., Casanova A., Romero A., Lio P., Bengio Y. Graph attention networks. // stat. 2017. Vol. 1050. N. 20.

[91] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Polosukhin I. Attention is all you need. // Advances in neural information processing systems. 2017. Vol. 30.