GRAPH NEURAL NETWORK BASED LEARNING FOR DYNAMIC SPECTRUM ACCESS

He Jiang
University of Rhode Island, jiang_he@uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation
https://digitalcommons.uri.edu/oa_diss/1245

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons@etal.uri.edu.
DOCTOR OF PHILOSOPHY DISSERTATION

OF

HE JIANG

APPROVED:

Dissertation Committee:

Major Professor       Haibo He
                      Yan (Lindsay) Sun
                      Lisa DiPippo
                      Brenton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2021
ABSTRACT

Over the past decade, we have witnessed tremendous technology development and societal benefits for various wireless devices, such as smartphones, smart wearable devices, and the Internet of Things (IoT), among others. To meet the traffic demands from these devices and provide a higher quality of experience (QoE) for users, wireless communication systems have exploited more and more radio spectrum. Although the capacity of the radio spectrum is fairly large, the shortage problem is imminent due to the dramatic proliferation of wireless devices. Dynamic spectrum access (DSA) is proposed as one of the most promising solutions to the spectrum shortage problem. The general objective of DSA is to allocate the spectrum resources to users dynamically and efficiently, which involves several control, coordination, and optimization problems. However, with the huge increment of wireless users, our wireless communication systems are much more complex than ever before and traditional DSA solutions can be low efficient or even impractical.

Recent breakthroughs achieved by the advanced neural network based machine learning algorithms have demonstrated their great potentials in solving complex large-scale problems. This dissertation focus on developing neural network based machine learning algorithms and models for problems involved in DSA. This work majorly depends on two machine learning techniques: graph neural network (GNN) and reinforcement learning (RL), which are utilized to solve different optimization, coordination, and control problems related to DSA, such as spectrum sensing, interference estimation, and dynamic spectrum allocation. The proposed algorithms and models are validated by numerous simulation studies.
ACKNOWLEDGMENTS

The over five years of study at the University of Rhode Island is challenging and memorable for me. At the end of this long journey, I would first express my great appreciation to my advisor, Prof. Haibo He. He has provided me numerous valuable suggestions, insights, and encouragement on my research work. From his guidance and advice, what I learned are not only the solutions to problems but also the ability to pursue the solutions logically and efficiently, which is an inestimable fortune to my life.

I also want to thank my committee members: Prof. Yan (Lindsay) Sun, Prof. Lisa DiPippo, and Prof. Marco Alvarez. Their comments and suggestions help me to improve my research work significantly. I truly appreciate their efforts and time spent on the preparation of my proposal, dissertation, and defense.

I also want to show my gratefulness to the group members I met in the Computational Intelligence and Self-Adaptive System (CISA) Laboratory: Bo Tang, Jing Wang, Yufei Tang, Xiangnan Zhong, Jun Yan, Zhiqiang Wan, Yazhou Zhang, Lusi Li, Zhenhua Wang, and Hepeng Li for their help on both work and life. I also appreciate every visiting scholars I met at the CISA lab for their friendships and supports.

I am deeply grateful for the support for my Ph.D. research from the National Science Foundation (NSF) under grants ECCS 1053717, CMMI 1526835, ECCS 1731672, and the Office of Naval Research under Grant N00014-18-1-2396. Their support provided me the opportunity to explore the important research topics in my Ph.D. career.
# TABLE OF CONTENTS

**ABSTRACT** ................................................................. ii

**ACKNOWLEDGMENTS** ..................................................... iii

**TABLE OF CONTENTS** .................................................... iv

**LIST OF FIGURES** ........................................................ vii

**LIST OF TABLES** ........................................................... xi

**CHAPTER**

1 *Introduction* ............................................................. 1

1.1 Research Motivations and Challenges ................................. 1

1.2 Graph Neural Networks .................................................. 2

1.3 Reinforcement Learning ................................................ 5

1.4 Research Objectives .................................................... 6

1.4.1 Graph Neural Network based Feature Extraction ............... 7

1.4.2 Energy-efficiency Cooperative Spectrum Sensing .............. 8

1.4.3 Interference Estimation for Device-to-Device Communications .. 9

1.4.4 Dynamic Spectrum Allocation for Femtocell Networks .......... 10

1.5 Dissertation Outline .................................................... 11

2 *Feature Extraction with Graph Neural Networks* .................... 13

2.1 Background .................................................................. 13

2.2 Graph Neural Network Based Semi-supervised Learning .......... 14

2.3 Negative Link Generation .............................................. 17
2.4 Graph Convolution with Negative Links .......................... 20
2.5 Proposed Training Framework ....................................... 21
2.6 Discussion .............................................................. 23
2.7 Experiment Results ..................................................... 26
   2.7.1 Experiment Settings ............................................... 26
   2.7.2 Node Classification Performance ................................. 27
   2.7.3 Model Analysis ..................................................... 29
2.8 Chapter Summary ....................................................... 31

3 Energy Efficiency Optimization for Cooperative Spectrum Sensing ............................................ 35
   3.1 Background ........................................................... 35
   3.2 System Model .......................................................... 37
   3.3 Energy-Efficient Distributed Cooperative Spectrum Sensing ........................................ 39
   3.4 Deep Reinforcement Learning Based Solution .................................................. 42
      3.4.1 Sequential Sensor Selection .................................... 42
      3.4.2 Proposed Model .................................................. 44
      3.4.3 Training with Reinforcement Learning ........................ 46
   3.5 Simulation Results ..................................................... 48
   3.6 Chapter Summary ....................................................... 51

4 Channel Interference Estimation for D2D networks .......................................................... 52
   4.1 Background ........................................................... 52
   4.2 System Model and Problem Formulation .................................. 54
   4.3 Estimating the Interference with Graph Convolutional Network .................. 58
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Model Training</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>Simulations and Analysis</td>
<td>63</td>
</tr>
<tr>
<td>4.6</td>
<td>Chapter Summary</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>Dynamic Spectrum Allocation Policy Design via Graph Neural Network</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>System Model</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>Graphical Channel Access Game</td>
<td>71</td>
</tr>
<tr>
<td>5.3</td>
<td>Dynamic Channel Access via Reinforcement Learning</td>
<td>72</td>
</tr>
<tr>
<td>5.4</td>
<td>Graph Neural Network based Implementation</td>
<td>73</td>
</tr>
<tr>
<td>5.5</td>
<td>Simulation Studies</td>
<td>78</td>
</tr>
<tr>
<td>5.6</td>
<td>Chapter Summary</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion and Future Work</td>
<td>86</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusion</td>
<td>86</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Work</td>
<td>88</td>
</tr>
</tbody>
</table>

LIST OF REFERENCES | 90 |

BIBLIOGRAPHY | 101 |
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A general illustration of DSA [1].</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Feature extraction procedure of a GNN.</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>The general agent-environment interaction process [2].</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>An overview of the dissertation organization.</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Illustration of the proposed negative link generation process.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The GCN estimates the classes for unlabeled nodes ($\tilde{y}<em>i$) based on the original graph and extracts the node representation vector ($z_i$) for each node. A fully connected neural network is used to calculate $\omega</em>{ij}$ and its input is set to the concatenation of the node representation vectors. The red squares reflect the magnitude of ${\omega_{ij}}$. And a square colored with dark red indicates a large value of the corresponding $\omega_{ij}$. A generated negative link is represented by the red dashed line, which specifies that the node at its head is a negative neighbor of the node at its tail.</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Boxplots of the magnitude statistics of generated negative links in every 10 consecutive training epochs for three datasets: (a) Cora, (b) Citeseer, (c) Pubmed. The box specifies the first quartile and the third quartile. The orange bar in the box shows the median. The whiskers under and above the box extend to the minimum and the maximum. The green triangle indicates the mean.</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>T-SNE embedding plot of the feature extracted by the first layer of the GCNs on three datasets: (a) Cora, (b) Citeseer, (c) Pubmed. The plots lie at the left column exhibit the node features extracted by the original GCN model of [3]. The plots lie at the right column exhibit the node features extracted by our model.</td>
<td>34</td>
</tr>
</tbody>
</table>
A sensor-aided cognitive radio network with energy detection based distributed cooperative sensing [4]: (a) sensor-aided cognitive radio network; (b) energy detection based distributed cooperative sensing. In the energy detection illustration, the node color indicates the local SNR level of the sensor measurement. Each node recursively communicates with its neighbor to fuse these measurements with a consensus protocol until convergence. The converged result is then compared with a pre-set threshold to infer whether the target spectrum band is idle or occupied.

Sequential sensor selection process for distributed cooperative sensing [4]. In this process, we pick one node a time until the summation of all the performance gains ($G$) of the selected nodes surpass a pre-set threshold ($\Theta$). In each selection step, the legitimate candidate nodes for selection are those connected to one of the previously selected nodes.

The proposed DRL framework [4]. The upper part depicts the decision making and training procedure. The lower part illustrates the specific structure of the graph value network, where Structure2Vec is the GNN used to generate the graph feature, and the fully connected neural network is used to calculate the action value.

Parameters of the considered sensor networks [4].

Performance comparison under different network scales [4]. In (a), x-axis denotes the number of nodes in the test network, and y-axis shows the average number of nodes selected by the three comparative approaches: GH (greedy heuristic) based approach, GA (genetic algorithm) based approach, and the proposed DRL (deep reinforcement learning) based approach. The performance for each network scale is based on the average of 1000 testing cases. In (b), the pie charts show the computation time (in seconds) of the GA based approach and the DRL based approach.

A D2D communication network. A transmitter directly communicate with the corresponding receiver. The D2D pairs are coordinated by the base station.

An illustration of the combination of $\mathcal{G}_R$ and $\mathcal{G}_{RT}$. 
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>An illustration of the combination of $G_T$ and $G_{RT}$</td>
<td>60</td>
</tr>
<tr>
<td>16</td>
<td>A D2D network containing 50 D2D pairs. The circles represent transmitters and the stars are the receivers. The paired D2D transmitter and receiver are connected by a solid line.</td>
<td>64</td>
</tr>
<tr>
<td>17</td>
<td>Estimation error during the training process of the network containing 50 initial D2D pairs: (a) average MSE of $\hat{I}<em>{k, i}$ achieved by our method and average estimation in 10 trials; (b) MSE of $\hat{I}</em>{k, i}$ achieved by our method in 10 trials where each line represents the result of one trial.</td>
<td>65</td>
</tr>
<tr>
<td>18</td>
<td>Estimation error during the training process of the network containing 100 initial D2D pairs: (a) average MSE of $\hat{I}<em>{k, i}$ achieved by our method and the average estimation in 10 trials; (b) MSE of $\hat{I}</em>{k, i}$ achieved by our method in 10 trials where each line represents the result of one trial.</td>
<td>66</td>
</tr>
<tr>
<td>19</td>
<td>An example of the femtocell network [5]. Each femtocell serves several users. The number in each femtocell is its index. Two APs can interfere with each other if their distance is less than the threshold.</td>
<td>70</td>
</tr>
<tr>
<td>20</td>
<td>The proposed graph neural network based architecture for the estimation of channel quality [5]. In this architecture, the first part is a graph neural network, which is used to extract a feature embedding for each vertex. GCN is applied for this part which generates the vertex feature through a local information aggregation process (also known as graph convolution). Then, based on the obtained feature, each node will evaluate the channel quality via a fully connected multi-layer neural network.</td>
<td>75</td>
</tr>
<tr>
<td>21</td>
<td>The communication topologies of the six femtocell networks [5]. The red circles in each subfigure denote the APs and the edges represent the interference relationship.</td>
<td>79</td>
</tr>
<tr>
<td>22</td>
<td>The evolution of the Q values of the six available channels [5]. The sub-figures (a)-(f) show the results of $C_1$-$C_6$ correspondingly. In a sub-figure, x-axis denotes the learning step, and the colored lines show $Q$ value evolution of all APs with one line representing an AP.</td>
<td>80</td>
</tr>
</tbody>
</table>
Figure 23 (a) The training error of the proposed network for the femtocell network with 30 APs. (b) The weights evolution process of the first convolutional layer of the applied GCN [5].

Figure 24 Performance comparison under different femtocell networks [5]. In (a), x-axis denotes the number of APs of the femtocell networks, and the height of the bar measured by y-axis announces the average interference level of the solution generated by the corresponding method. This results for each femtocell network is based on the average of the 2000 testing cases. In (b), the pie charts show the computation time (in seconds) of the autonomous best response algorithm and the proposed graph neural network based reinforcement learning approach.
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dataset Statistics</td>
<td>26</td>
</tr>
<tr>
<td>2</td>
<td>Classification Accuracies (%) with the Standard Data Split. The <strong>Bold Marker</strong> Denotes the Best Performance on that Dataset, and the Star Marker (⋆) Means the Best Algorithm Pasts T-test Compared with the Second-best Algorithm with the Significance Level of 0.001. The (±) error bar denotes the standard deviation and for some of the methods, this term is omitted for not being reported by the original literature.</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>Classification Accuracies (%) with Validation Set being Utilized for Training. The <strong>Bold Marker</strong> and the Star Marker (⋆) Are Consistent with the Definitions in Table 2</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Classification Accuracies (%) with Reduced-label Training. The <strong>Bold Marker</strong> and the Star Marker (⋆) Are Consistent with the Definitions in Table 2</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Comparison of Node classification Accuracies (%) of Our Method under Different Feature Propagation Model</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Comparison of Node Classification Accuracies (%) with Self-training</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>System parameters</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>Multiagent Q learning for the channel access game [5]</td>
<td>74</td>
</tr>
<tr>
<td>9</td>
<td>The training process of the proposed model [5]</td>
<td>77</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

1.1 Research Motivations and Challenges

In recent years, the number of various wireless devices has greatly increased, which includes smartphones, tablets, wearable devices. With a vast population of wireless devices, it is necessary to upgrade our wireless communication networks to match the unprecedented traffic demands from these wireless devices. Traditionally, this objective can be achieved by utilizing more spectrum or deploying more macro base stations [6]. However, the available spectrum is quite limited, and the deployment cost of the macro base station is usually high. Currently, most spectrum resources are exclusively allocated to licensed users. However, actual spectrum usage measurements obtained by the FCC’s Spectrum Policy Task Force [7] indicates that at any given time and location, much of the prized spectrum lies idle, which is a big extravagance of this scarce natural resource. In contrast, DSA [8, 9, 10], which shares the spectrum resources among users and flexibly assign the spectrum resources based on the user’s demands, has great potentials to improve the spectrum efficiency and alleviate the spectrum shortage problem. Fig. 1 shows the basic idea of DSA.

Successful implementation of DSA can improve the spectrum efficiency by utilizing idle spectrum resources. However, the implementation of DSA is more complex than its static counterpart. Several techniques need to be conquered, such as the detection of idle spectrum [11, 12, 13], the sharing of the spectrum opportunities [14, 15], and interference management [16, 17]. DSA can be implemented for different communication system models, such as cognitive radio (CR) [18, 19, 20, 21], device-to-device (D2D) networks [22, 23, 24, 25, 26], and femto-cell/small cell networks[27, 28, 29, 30, 31]. For different system settings and as-
assumptions, various optimization, coordination, and control problems are involved. Most of the traditional methods for these problems search the corresponding solutions through model-based optimization techniques or iterative searching processes [32, 33, 28, 34, 31, 35, 36]. However, with a large population of users where a large portion can be mobile users, a modern large-scale communication system can be extremely complex and difficult to model. Therefore, traditional DSA solutions can be low-efficient or even invalid.

Recent breakthroughs of neural network based machine learning models have achieved remarkable successes across a wide range of application domains, such as computer games, natural language processing, pattern recognition, and medical diagnosis, to name a few, which demonstrates the power of machine learning in solving complex problems. This dissertation aims to develop machine learning based solutions for different problems of DSA. Among other efforts towards this objective, the proposed models and algorithms majorly depend on two machine learning techniques: GNN and RL, which are introduced next.

1.2 Graph Neural Networks

When applying machine learning techniques to a complex system, the first necessary step is usually extracting features from the system states as the system
state representations. Recently, neural networks are widely employed for this purpose [37, 38, 39, 40, 41]. However, in most of the previous works, the objective system structure or the system state dimension is fixed. For instance, in [38], the original system state is the image of the video game; and in [39] the system state is a vector of fixed length. However, for real-world systems, such as power systems, social networks, and communication networks, the system structure can be irregular and the topology and dimension can vary due to the mobility of the users/device.

Graph is a widely used model to analyze such systems as it can effectively reveal its intrinsic nature of local interaction [42, 43, 44, 45, 46, 47, 48, 49, 50]. And GNN is proposed to solve graph-related learning problems in a variety of disciplines. GNN is a generalized neural network architecture designed for processing graph-structured data with fixed or varied topologies. GNNs are demonstrated to be excellent feature extractors for graphs by their performance over a wide range of graph mining problems [51, 52, 53]. A general feature extraction process of GNN can be expressed by the following equations:

\[
\begin{align*}
H^{(0)} &= X \\
H^{(l)} &= \sigma(\hat{A}H^{(l-1)}W^{(l)})
\end{align*}
\]

(1)

where \(X\) is the initial feature of the graph data, \(\hat{A}\) is the “normalized” adjacency matrix; \(H^{(l-1)}\) is the feature matrix of layer \(l - 1\); and we have \(H^{(0)} = X\); \(W^{(l)}\) is a trainable weight matrix; and \(\sigma(\cdot)\) is an element-wise nonlinear activation function.

Eq. (1) can be decomposed into three consecutive steps: feature propagation, linear transformation, and nonlinear activation [54], which are illustrated by Fig. 2. At the feature propagation step, the feature vector of each node is smoothed by the features of its neighbors, where the coefficients of the neighbors are controlled by \(\hat{A}\). Then, the smoothed node features are transformed to a new feature space through the weight matrix \(W^{(l)}\). Finally, a nonlinear element-wise activation function, \(\sigma(\cdot)\),
is applied and the output serves as the input of the next layer.

Popular models include graph convolutional network (GCN) [3] and graph attention networks (GAT) [55]. Compared with the label similarity assumption practiced by the traditional graph Laplacian based methods [56, 57, 58], GNN-based approaches extend the similarity assumption to node features. This idea is reflected by the way of the feature inference procedure of GNNs, where the feature vector of one node is updated by the weighted summation of its neighbors’. This process smooths the node feature based on its neighborhood and the coefficients of the neighbors can be determined via different strategies. For instance, with the GCN model of [3], the coefficients of the nodes are completely calculated according to the node degrees, while some other researchers believe that the neighbors of a node possess different importances so that an attention mechanism is introduced to adaptively assign the coefficients based on the node features [55]. Since wireless communication networks can be modeled as a graph, GNN techniques have great
application potential on related optimization and control problems, which is a key technique of this dissertation.

1.3 Reinforcement Learning

RL [2] is a general learning framework that can automatically search optimal solutions for various optimization problems. One of the most distinguished properties of RL is that the optimal solutions can be found without the knowledge of the objective system or environment, and the learning is implemented in a data-driven style[59]. RL has been applied to numerous areas, such as optimal control [60, 61], finance investment [62], smart grid [63, 64], social behaviors [65, 66], and communication networks [67, 68, 69, 70, 71], just to name a few. In general, RL concerns how an agent learn the optimal actions/control policies of a Markov decision process (MDP) from the interactions with the environment. The agent-environment interaction in a reinforcement learning process can be illustrated by Fig. 3. At

\[ a^t = \pi(s^t) \]  \hspace{1cm} (2)

where \( \pi(\cdot) \), also known as the policy, is a mapping from the environment state to the action space. Then, the environment transits to the next state \( s^{t+1} \) and returns a feedback reward \( r^{t+1} = r(s^t, a^t, s^{t+1}) \) to the agent. Under policy \( \pi \) and

Figure 3. The general agent-environment interaction process [2].
state \( s \), the expected discounted cumulative rewards over an infinite time horizon is defined as
\[
V_{\pi}(s) = E\left(\sum_{i=0}^{\infty} \gamma^i r^{t+i+1} | s^t = s \right).
\]

In reinforcement learning, the objective of an agent is to find an optimal policy to maximize the cumulative rewards. Let the action value, or the Q value, be defined as
\[
Q_{\pi}(s, a) = E_{\pi}\left[\sum_{i=0}^{\infty} \gamma^i r^{t+i+1} | s^t = s, a^t = a \right]
\]
which represents the expected discounted rewards of taking action \( a \) under state \( s \). Given all action values of a certain state, the optimal policy should be selecting the action with the highest expected returns. Therefore, the optimal policy can be expressed by
\[
\pi^*(s) = \arg\max_a \{Q_{\pi^*}(s, a)\}.
\]
Additionally, the expected cumulative rewards of the optimal policy under state \( s \) is
\[
V_{\pi^*}(s) = \arg\max_a Q_{\pi^*}(s, a).
\]
Furthermore, we can obtain the relation of \( Q_{\pi^*} \) and \( V_{\pi^*} \):
\[
Q_{\pi^*}(s, a) = \sum_{s'} P(s' | s, a) r(s, a, s') + \sum_{s'} P(s' | s, a) V_{\pi^*}(s')
\]
where \( P(s' | s, a) \) represent the state transit function.

1.4 Research Objectives

Given the challenges residing in the DSA of wireless communications and inspired by the recent successes achieved by machine learning models, this study defines the following objectives: GNN based feature extraction, energy-efficiency cooperative spectrum sensing, interference estimation for device-to-device communications, and dynamic spectrum allocation for femtocell networks.
1.4.1 Graph Neural Network based Feature Extraction

Neural networks are widely used to learn general system representations for various applications. However, for a wireless communication system, as the system structure is non-regular and can change with the movement of the users, traditional neural networks, e.g., fully connected neural networks [72] and convolutional neural networks (CNNs) [73], cannot be applied. Since a communication network can be modeled as a graph [33, 42, 74, 43], GNNs will be used in this work to extract system features. To better understand the mechanism of the GNNs, this work will investigate the GNN-based representation learning under the semi-supervised node classification framework, which is one of the most popular approaches to develop GNN models. Currently, most GNN models are designed based on the similarity assumption that neighboring nodes are likely to share similar features. This idea is reflected by the way of the feature inference procedure of GNNs, where the feature vector of one node is updated by the weighted summation of its neighbors’. This process smooths the node feature based on its neighborhood and the coefficients of the neighbors can be determined via different strategies. For instance, with the GCN model of [75], the coefficients of the nodes are completely calculated according to the node degrees, while some other researchers believe that the neighbors of a node possess different importances so that an attention mechanism is introduced to adaptively assign the coefficients based on the node features [55].

In most of the prior works, the GNN-based methods are designed to deal with the graphs only containing positive links that are used to model the positive correlations of connected nodes. This dissertation assumes negative correlations among nodes also exist in a graph, which can be exploited to better infer node features or labels. This dissertation will propose a novel learning framework that can generate meaningful negative links to describe inverse correlations of node
pairs. A negative link is directed and specifies that the node at its head is the negative neighbor of the node at its tail. And it is associated with a negative weight to quantify the negative correlation. Furthermore, a GNN model that is compatible with graphs containing negative links as well as a corresponding optimization algorithm to enable the generated negative links to improve the semi-supervised learning performance will also be developed.

1.4.2 Energy-efficiency Cooperative Spectrum Sensing

Cognitive radio is one of the most promising DSA schemes to boost the radio spectrum efficiency [19]. In a cognitive radio network, the spectrum bands are allowed to be accessed by both licensed and unlicensed users, which are also called primary users (PUs) and secondary users (SUs), respectively. The PUs are empowered with priority to use the spectrum, whereas the SUs are permitted to access the spectrum only if the PUs are inactive. Under this regulation, spectrum sensing plays a crucial role as the detection procedure for the PUs’ activities. Inherently, spectrum sensing belongs to the signal detection realm. Various classic detection techniques can be applied to identify the presence of the PUs. However, the performance of a single detector can be severely degraded in the real-world environment because of shadowing, multipath fading, and hidden terminal issues [13]. In this case, cooperative spectrum sensing is proposed to improve the detection performance by taking advantage of the spatial diversity of the distributed sensors. According to the communication topology, the cooperative sensing paradigms fall into three categories: centralized, relay-assisted, and distributed [13]. For the first two categories, a fusion center is necessary to collect the sensing data from the sensors and make the detection decision with a fusion rule. In this way, the fusion center can suffer from severe traffic overload. By contrast, distributed cooperative sensing accomplishes the collaboration through a local iterative consensus
algorithm, where each sensor only processes the data from its neighbors [35, 36].

Regardless of which cooperative sensing paradigm is used, the signal detection
devices are indispensable to collect and process the sensing data. Traditionally,
the spectrum sensing module is integrated with every SU, which will increase the
manufacturing cost of the wireless devices. As a result, several researchers advocate
separating this functionality from the SUs and constructing a specialized sensor
network to provide the sensing service [76]. The sensor network is supposed to
operate continuously to provide a real-time spectrum map for SUs. Consequently,
energy consumption can be considerable over time. This issue could be more severe
for a distributed cooperative sensing system since it relies on an iterative algorithm.

This work will develop a GNN-based model for topology optimization of the
spectrum sensor network. The proposed method optimizes the sensor network
through a sequential decision-making process. Reinforcement learning is applied
to train the model to learn a generalized policy that can be applied to the systems
of different scales and structures.

1.4.3 Interference Estimation for Device-to-Device Communications

D2D communication enables mobile users nearby to directly communicate
with each other, which can bring about many benefits, e.g., high data rate, low
power consumption, and lower delays due to the proximity of the transmitter and
receiver [77, 26]. Moreover, D2D communication can also improve the spectrum ef-
ficiency since the low transmission power makes the spectrum spatial reuse possible
[78].

In a D2D communication network, the spectrum resources are shared by nu-
merous D2D users. As a result, a D2D link can suffer from not only the local noise
but also complex co-channel interference incurred by other D2D users. Therefore,
interference mitigation schemes are intensively studied from different perspectives
Generally, for an interference management policy, the interference information of each D2D user is essential. This work will develop a GNN-based model for the interference estimation for the users of wireless communication networks. The interference information is two-fold, which contains the interference from all other existing D2D links to the new receiver and the potential interference introduced by the new transmitter to other D2D links. The D2D network change dynamically, which means a D2D pair will terminate the D2D link when they finish the data transmission and new D2D communication requests can occur over time.

1.4.4 Dynamic Spectrum Allocation for Femtocell Networks

Femtocell network is proposed as a cost-effective solution to provide sufficient traffic capacity [6, 27, 83]. In a femtocell network, several access points (APs) are spatially deployed, and the wireless users can access the wireless network via the AP nearby. In this way, the distance between the transmitter and the receiver can be greatly reduced, which will bring multiple benefits, such as more reliable communications, higher throughputs, power/battery savings, and so forth [27]. A significant property of the femtocell network is that the transmitting power of an AP is low, such that it is possible to boost the spectrum efficiency via spectrum spatial reuse [84]. Moreover, the traffic loads of the femtocells can be varied both temporally and spatially [85]. Consequently, DSA [10] for femtocell network attracts the interests of numerous researchers in the community. Substantial dynamic spectrum access methods have been proposed to improve the spectrum efficiency from different aspects [32, 33, 28, 34, 31].

Most of these previous works obtain their spectrum access policy via an iterative algorithm, which can be inefficient for a communication system with dynamic traffic loads since the involved iteration process need to be reimplemented for each
encountered traffic load.

This dissertation will design a GNN based model for dynamic spectrum allocation. This model is able to learn a mapping function that directly maps the traffic load of each wireless user to the spectrum access policy and the obtained spectrum access policy should effectively reduce the interference among the wireless users. Besides, the learned spectrum access policy should be able to adapt to different system states without retraining the mapping function.

1.5 Dissertation Outline

The general organization of the rest parts of this dissertation is shown in Fig. 4 Chapter 2 presents the research on GNN-based graph representation learning. A

![Diagram showing the organization of the dissertation](image)
novel learning framework that improves the graph feature inference capability by considering negatively correlated neighbors is designed. Chapter 3 focuses on the energy efficiency optimization problem for the cooperative spectrum sensing of cognitive radio. And an end-to-end learning framework that utilizes GNN and RL is designed to search the energy-efficient cooperative spectrum sensing schemes. Chapter 3 formulates a channel interference estimation problem for D2D communication networks. A GNN-based solution is proposed to obtain accurate estimations for channel interference. Chapter 5 studies the DSA for femtocell networks. A GNN-based model is developed to generate a high-efficient spectrum allocation policy and an RL-based training algorithm is designed to optimize the trainable parameters of this model. The conclusion and future work are with Chapter 6.
CHAPTER 2

Feature Extraction with Graph Neural Networks

2.1 Background

Various GNN variants have been proposed to learn graph representations and have achieved tremendous successes across a wide range of graph mining problems [51, 52, 53]. Most of the prior works for GNN-based graph representation learning are investigated under the semi-supervised learning framework [3, 56, 75]. This chapter will first formulate the semi-supervised learning problem. GNN-based approaches depend on the similarity assumption that neighboring nodes are likely to share similar features. This idea is reflected by the way of the feature inference procedure of GNNs, where the feature vector of one node is updated by the weighted summation of its neighbors’. This process smooths the node feature based on its neighborhood and the coefficients of the neighbors can be determined via different strategies.

Numerous algorithms have been proposed to improve the performance of GNNs. Adversarial training algorithms developed by [86, 87, 88] aims to utilize perturbations on node features to improve the robustness of GCNs. In [86], a virtual adversarial loss is built based on the KL divergence between node label distribution calculated from the original graph and the distribution calculated from the graph with node perturbations. This loss is used as a regularization term of the overall training loss function to improve the generalization performance of GNNs. Besides the virtual adversarial loss, Feng et al. [87] include a graph adversarial regularizer that encourages the graph adversarial examples to be classified similarly to the connected examples. An adversarially regularized graph embedding learning algorithm is proposed by [88], which aims to learn latent node codes that preserve graph topology information. In [89], a variational expectation-maximization algo-
rithm is developed to enable the interactive learning of two GNNs with one learning node representation and the other modeling label dependencies. To learn disentangled node features, a neighborhood routing mechanism is proposed to assign the neighbors to different channels based on the underlying factors [90]. There also exist some works that deem the observed graph topology as a sample of the distribution that models the probability of node connectivity [91, 92]. And, generative models are employed to generate different graph topologies to train GNNs, such that the classification performances are improved, especially when the training samples are very few.

In this chapter, we develop a novel GCN-based learning framework that improves the node representation inference capability by including negative links in a graph. Negative links in our method define the inverse correlations for the nodes connected by them and are adaptively generated through a neural network based generation model. This chapter also designs a joint optimization learning framework to make the generated negative links beneficial for the classification performance.

2.2 Graph Neural Network Based Semi-supervised Learning

Let a graph be denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is a finite set of nodes; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the link set. The link $e_{ij}$ is represented by an ordered node pairs $(v_i, v_j)$. Additionally, the weights of links are specified by a symmetric adjacency matrix $A \in \mathbb{R}^{N \times N}$. The entry lie in row $i$ and column $j$, i.e., $A_{ij}$, is the weight of $e_{ij}$. Generally, $A_{ij} > 0$ if $(v_i, v_j) \in \mathcal{E}$; otherwise, $A_{ij} = 0$. If $e_{ij}$ exists, $v_j$ is called a neighbor of $v_i$. The set of neighbors of $v_i$ is denoted by $\mathcal{N}_i$. For $v_i$, there is a feature vector $x_i$ and a label $y_i \in \{1, 2, \ldots, C\}$ representing its class, where $C$ is the total number of the classes. In a semi-supervised node classification problem, only a subset of nodes have the label information. Without loss of generality, we
assume the first $M$ nodes, $\{v_1, \ldots, v_M\}$, are labeled. Let $\mathcal{F}$ denote the set of $N \times C$ matrices with non-negative entries. The one hot label matrix of all nodes is defined as $Y \in \mathcal{F}$, where $Y_{ij} = 1$ if $v_i$ is labeled as $y_i = j$; otherwise, $Y_{ij} = 0$. The learning objective is to predict the classes of all unlabeled nodes. For this problem, several graph Laplacian regularization based approaches are proposed, which depends on the assumption that two connected nodes are likely to share a common label [57, 58]. For instance, in [57], the cost function is designed as

$$Q(F) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i} A_{ij} \| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \|_2^2 + \mu \sum_{i=1}^{N} \|F_i - Y_i\|^2.$$  

(8)

In the equation above, $F = [F_1, \ldots, F_N]^T$; the $C$-dimensional vector $F_i$ is the classification of $v_i$ by labeling $y_i = \arg \max_{j \leq C} F_{ij}$; and $\mu$ is a weighing factor. The first term of the right-hand side of Eq. (8) provides the Laplacian regularization. By minimizing this term, the classification of the connected nodes is smoothed over the graph. Here, graph links are considered as descriptions of class similarities for the connected nodes. Recently, some researchers believe that the graph links can convey additional information besides the class similarities. As a result, GNN-based methods are proposed, which extend the similarities of the connected nodes from labels to features, and have achieved significant breakthroughs.

A GNN based feature extraction model obtains the meaningful node features through a stack of graph convolutional layers. Denoting node features at layer $l$ by $H^{(l)} \in \mathbb{R}^{N \times K^{(l)}}$ where $K^{(l)}$ is the feature dimension, the computation formula of $H^{(l)}$ is

$$H^{(l)} = \sigma (\hat{A} H^{(l-1)} W^{(l)})$$  

(9)

where $\hat{A}$ is the “normalized” adjacency matrix; $H^{(l-1)} \in \mathbb{R}^{N \times K^{(l-1)}}$ is the feature matrix of the previous layer; and we have $H^{(0)} = X$ with $X = [x_1, \ldots, x_N]^T$;
where $\hat{A} = A + I_N$; $\hat{D}$ is the degree matrix of $\hat{A}$; and $I_N$ is the $N \times N$ identity matrix. In this way, $\hat{A}$ is completely determined by the adjacency matrix. Veličković et al. [55] consider the nodes within a neighborhood to be of different importances, and they propose an adaptive attention mechanism to learn the importances of the neighbors. Some researchers assume that the original graph topology is not enough to describe the node dependency. Accordingly, they deem the topology of a graph as a random variable and learn the corresponding distribution through graph generative models [91, 92]. Besides architecture innovations, modifications on training algorithms are also demonstrated to be beneficial to the performance of GNNs [86, 87, 89].

Most of the previous GNN-based methods locally smooth the node feature by assigning the neighbors with positive coefficients, where a node is assumed to be positively correlated with its neighbors. In this chapter, a novel GNN learning
framework is proposed, which improves the feature and label inference capability by considering the nodes of inverse correlated properties, called negative neighbors. As the original graph contains positive links only, this chapter first proposes a negative link generation model. Then, the GCN model designed for graphs with positive links [3] is modified to be compatible with the graph with negative links.

2.3 Negative Link Generation

To generate a negative link for a node, two questions need to be answered: 1) which node is selected as the negative neighbor, and 2) what is the weight of the negative link. Consequently, our proposed negative link generation model aims to solve these two questions for each \( v_i \). The generation procedure is summarized by the following equations:

\[
\text{for } j = 1, \ldots, N
\]

\[
\omega_{ij} = \begin{cases} 
-\infty & \text{if } y_i = y_j \\
 f_\theta(z_i, z_j) & \text{otherwise}
\end{cases}
\]  \hspace{1cm} (11)

\[
i_c^{(-)} = \arg \max_j \{ \omega_{ij} | y_j = c \}, c = 1, 2, \ldots, C
\]  \hspace{1cm} (12)

\[
i^{(-)} = \arg \max_{i_c^{(-)}} \{ \omega_{ii^{(-)}} \}_c^{C}
\]  \hspace{1cm} (13)

\[
A_{ii^{(-)}} = -\frac{\exp(\omega_{ii^{(-)}})}{\sum_{c=1}^{C} \exp(\omega_{ii^{(-)}})}
\]  \hspace{1cm} (14)

In Eq. (11), \( \omega_{ij} \) is a quantity measuring the potential of \( v_j \) being a negative neighbor for \( v_i \); \( z_i \) and \( z_j \) are the feature representations of \( v_i \) and \( v_j \); \( f_\theta \) is a trainable function parameterized by \( \theta \); and \( \omega_{ij} \) is constrained to be negative infinity if \( v_i \) and \( v_j \) belong to the same class, which means negative neighbors are only searched from different classes. Eq. (12) finds the index of the nodes with highest \( \omega_{ij} \) in each class, which is denoted by \( i_c^{(-)} \) for class \( c \). \( v_i^{(-)} \) is considered as the representative of the potential negative neighbors in class \( c \). Among the \( C \) representative nega-
tive neighbors (one for each class), the final negative neighbor \(v_{i\ell(-)}\) is selected by Eq. (13). The generated negative link connecting \(v_i\) and \(v_{i\ell(-)}\) is denoted by \(e_{i\ell(-)}\), which is directed, \(i.e.,\) from \(v_i\) to \(v_{i\ell(-)}\). And the associated weight of this negative link \(A_{i\ell(-)}\) is calculated via Eq. (14), where a softmax function is implemented over \(\{\omega_{i1(-)},\ldots,\omega_{iC(-)}\}\). In this way, other representative negative neighbors are used as references to assess the connecting strength of \(e_{i\ell(-)}\). And the negative sign at the right hand side of Eq. (14) indicates \(v_{i\ell(-)}\) is negatively correlated to \(v_i\). Filling \(A\) at row \(i\) column \(i\ell(-)\) with \(A_{i\ell(-)}\) for \(i = 1, \ldots, N\), the updated adjacency matrix is denoted by \(A^*\) which can be asymmetric because the negative links are directed. It should be noted that a generated negative link can possibly connect two nodes that have already been connected with positive links, and if so, the existing entry of \(A\) is substituted with \(A_{i\ell(-)}\). The reason for this operation is two-fold. First, the positive links are usually built based on the modeling strategy, and GCN-based method use positive links to describe the similarities of the connected nodes; however, this relationship can be noisy which is also point out by [91, 92]. Second, in the proposed method, the optimization algorithm, which is introduced in the next section, makes the generated negative links beneficial for the node feature inference so that we will substitute the positive link if it is conflicting with a generated negative link.

According to Eq. (11) and (12), the proposed negative generation process requires node labels, which are not available for the nodes outside the training set. Under this situation, a GCN is employed as an inference or recognition model to provide an estimation of the class for the unlabeled nodes. This GCN is denoted by \(g_\psi(X, A)\) where \(\psi\) is the weights; \(X\) is the node feature matrix; and \(A\) is the adjacency matrix. Moreover, this GCN can also be used to extract node representations simultaneously. An implementation scheme of the proposed negative
The generated negative links are expected to be beneficial for the inference capability of \( g_\psi \). Naturally, the generated negative links should be evaluated through...
that requires \( g_\psi \) to be compatible with \( A^* \). However, as \( A^* \) is asymmetric and contains negative elements, most of the previous GCN models, designed for graphs with positive symmetric adjacency matrices, cannot be applied directly. Thus, in the remaining part of this section, we present the modifications on the GCN model of [3] to make it compatible with both the original graph and the graph with negative links.

### 2.4 Graph Convolution with Negative Links

The proposed GCN model is developed by modifying the GCN introduced by [3]. The feature updating rule at layer \( l \) of the modified model is described by

\[
H^{(l)} = \hat{A}^* \sigma(H^{(l-1)}W^{(l)})
\]

\[
\hat{A}^* = \tilde{D}_{\text{out}}^{-1/2} A^* \tilde{D}_{\text{in}}^{-1/2}
\]

where \( \tilde{A}^* = A^* + I_N \); \( \tilde{D}_{\text{in}} \) and \( \tilde{D}_{\text{out}} \) are the ‘modified’ indegree matrix and outdegree matrix of \( A^* \). \( \tilde{D}_{\text{in}} \) and \( \tilde{D}_{\text{out}} \) are diagonal matrices and the \( i \)-th diagonal elements of these two matrices are \( (\tilde{D}_{\text{in}})_{ii} = \sum_j |A^*_{ji}| \) and \( (\tilde{D}_{\text{out}})_{ii} = \sum_j |A^*_{ij}| \), respectively. Here, we use the term ‘modified’ because with the original definition in graph theory, the indegree matrix and the outdegree matrix are calculated based on the value of the entries of an adjacency matrix rather than the absolute value.

Eq. (16) defines the specific form of \( \hat{A}^* \), which concerns the coefficients assigned to the neighbors when smoothing node features. When smoothing the feature vector of \( v_i \), the coefficients of its neighbors are specified by the \( i \)-th row of \( A^* \) and we have

\[
h_i \leftarrow \frac{1}{\sqrt{(\tilde{D}_{\text{out}})_{ii}}} \left( \frac{h_i}{\sqrt{(\tilde{D}_{\text{in}})_{ii}}} + \sum_{j \in N_i} A^*_{ij} \frac{h_j}{\sqrt{(\tilde{D}_{\text{in}})_{jj}}} \right).
\]

In this procedure, a node sends out its feature information through both negative links and positive links. The quantity of the information sent out along a link is
proportional to the magnitude of this link. Thus, the indegree of a node measures how much information is sent to the other nodes. To make the nodes with different indegrees have a comparable influence on the whole graph, the feature vectors are first normalized by the corresponding node indegrees. This normalization is important for the graph with generated negative links since a node may be the negative neighbor of many other nodes. Then, a node $v_i$ aggregates the normalized features from both positive neighbors and negative neighbors. To keep the scale of the feature vector, the aggregated feature information is normalized by the node outdegree before being used to update $h_i$.

Eq. (15) specifies the feature transformation procedure from layer $l-1$ to layer $l$. With the feature transformation designed by [3], namely, Eq. (9), $\hat{A}$ is multiplied with $H^{(l-1)}$ first, which means the input of each layer is smoothed. However, in our method, we smooth the node feature at the feature space of the output. In this way, the GCN output of one node is directly affected by the neighbors and the effects of negative links can be more explicitly identified.

Eq. (16) can be deemed as a generalized form of Eq. (10) because when applying Eq. (16) to a graph with no negative link, $\hat{A}^*$ will automatically reduce to $\hat{A}$. Therefore, the modified GCN is also applicable to the original graph that contains no negative links.

2.5 Proposed Training Framework

With the proposed model, the label distribution for each node can be estimated based on the original graph as well as the graph with generated negative links. The class distribution calculated by $g_{\theta}(X,A)$ is denoted as $\hat{Y} \in \mathcal{F}$ with $\hat{Y}_{ic}$ being the probability of $v_i$ belonging to class $c$. And the class probability matrix is written as $\hat{Y}^* \in \mathcal{F}$ if it is calculated by $g_{\phi}(X,A^*)$. Note that $\hat{Y}^*$ depends on both $\psi$ and $\theta$ while $\hat{Y}$ is only related to $\psi$. 

21
The cross-entropy loss of $\hat{Y}$ and $\hat{Y}^*$ over the labeled samples is
\[
\ell_1(\theta, \psi) = - \sum_{i=1}^{M} \sum_{c=1}^{C} Y_{ic} \log \hat{Y}_{ic} - \sum_{i=1}^{M} \sum_{c=1}^{C} Y_{ic} \log \hat{Y}_{ic}^*.
\] (18)

By minimizing $\ell_1(\theta, \psi)$ using the gradient for $\theta$, the negative link generation model is tuned to assign large weights to the negative links that are beneficial for reducing the supervised loss. When updating $\psi$ based on $\ell_1(\theta, \psi)$, the inference function $g_\psi(X, A)$ is tuned to make more accurate prediction on the labeled nodes. Meanwhile, $\psi$ is also trained to deal with the graph with negative links since the supervised loss of $\hat{Y}^*$ provides the gradient for $\psi$ as well. Besides Eq. (18), we also add the following cross entropy between the $Y^*$ and $Y$ over the unlabeled samples as the other part of the training loss:
\[
\ell_2(\theta, \psi) = - \sum_{i=M+1}^{N} \sum_{c=1}^{C} \hat{Y}_{ic} \log \hat{Y}_{ic}^*.
\] (19)

The significance of Eq. (19) can be explained from two aspects. First, with $\ell_2(\theta, \psi)$, the utilization efficiency of the generated negative links can be largely improved. If the training loss is set to $\ell_1(\theta, \psi)$, only a small part of the generated negative links can be utilized for model training, which are the ones within the receptive field of the labeled nodes, while if $\ell_2(\theta, \psi)$ is also included, all the generated negative links participate in the optimization since $\ell_1(\theta, \psi)$ and $\ell_2(\theta, \psi)$ jointly cover all nodes in the graph. Second, $\ell_2(\theta, \psi)$ enables the inference function $g_\psi(X, A)$ to learn from the generated negative links. Minimizing $\ell_2(\theta, \psi)$, the difference between $\hat{Y}$ and $\hat{Y}^*$ are reduced. In this process, the inference function $g_\psi(X, A)$ is trained to generate similar label distribution to $\hat{Y}^*$ which is inferred based on the graph containing negative links.

In a semi-supervised learning problem, there are usually many more unlabeled nodes than the labeled nodes such that $\ell_2(\theta, \psi)$ can be considerably larger than
\ell_1(\theta, \psi). Therefore, the overall loss function of our model is set to

\[ \mathcal{L}(\psi, \theta) = \frac{1}{M} \ell_1(\psi, \theta) + \frac{\alpha}{N - M} \ell_2(\psi, \theta) \]  

(20)

where \( \ell_1(\psi, \theta) \) and \( \ell_2(\psi, \theta) \) are normalized by the involved node numbers to make them comparable; and \( \alpha \) is a hyperparameter that controls the portions of these two terms. With the loss function of Eq. (20), the model can be trained by the backpropagation algorithm. However, when generate \( A^* \), the generation process of Eq. (11)-(14) searches the negative neighbors for one node from the whole graph, which can be of low efficiency. To reduce the computation complexity, we set the search space to a small subset of the nodes rather than the whole graph. Specifically, in the negative link generation process, after obtaining the estimation of the node classes, we randomly sample \( T \) nodes \((T \ll N)\) in each class, which are used as the potential negative neighbors for that class. The reason for this operation is that there are no ground-truth negative neighbors for one node, and it is unlikely to find a very special negative neighbor by searching the whole graph. Consequently, the optimization procedure of our model can be summarized by Algorithm. 1. Note that at the early stage of the algorithm, it is possible that the nodes been assigned to a class is less than \( T \) because \( g_\psi \) is not well trained and can generate very biased classification. In this case, we duplicate the nodes in that class until the total number is greater than \( T \) such that the consistency of our algorithm can be kept.

2.6 Discussion

In this work, the final predictions of the node labels are provided by the inference model \( i.e., g_\psi(X, A) \), which takes the original graph as input, and the generated negative links are used to provide extra information to improve the performance of the inference function. Similar approaches can be found in prior works, such as\[89\] and \[92\]. In \[89\], the proposed model contains two GCNs with one in-
Algorithm 1 Training Algorithm

**Input:** $A, X, \{y_i\}_1^M$

**Parameter:** $\psi, \theta$

**Output:** Prediction of $\{y_i\}_M^{N+1}$

1. **while** training() **do**
   2. Calculate $\hat{Y}$ through $g_\psi(X, A)$
   3. Sampling $\{y_i\}_i=M+1^N$ based on $\hat{Y}$
   4. Uniformly sampling $T$ nodes in each class as potential negative neighbors
   5. Generate $A^*$ through Eq. (11)-(14)
   6. Calculate $\mathcal{L}(\psi, \theta)$ through Eq. (18)-(20)
   7. Update $(\psi, \theta)$ with $(\frac{\partial \mathcal{L}(\psi, \theta)}{\partial \psi}, \frac{\partial \mathcal{L}(\psi, \theta)}{\partial \theta})$
8. **end while**

9. Calculate $\hat{Y}$ with $g_\psi(X, A)$
10. $\hat{y}_i = \arg \max_{j \leq C} \hat{Y}_{ij}^i$ for $i = M + 1, \ldots, N$
11. **return** $\{\hat{y}_i\}_{M+1}^N$

ferring the node labels based on the node features and the other one providing the label dependencies that is utilized to improve the performance of the inference model. In [92], a link generation model is proposed where a fully connected neural network is used to generate labels for each node at first, and then the node features and the generated labels are together employed to generate the links. This process is trained to maximize the likelihood of the observed links and minimize the likelihood of the nonexistent links. With the approach of [92], a GCN-based recognition model is used to predict the labels of the nodes, which is trained based on the labeled nodes as well as the node labels generated by the fully connected neural network of the link generation process.

The proposed negative link generation process creates one negative link for each node, which is conservative. The current work does not generate multiple negative links for one node since there is no prior statistic model for the number of negative links and it is relatively safe to just generate the one that we are most confident about. Although experiment results in Section 2.6 demonstrate that the proposed method can achieve state-of-the-art performance in most settings,
a generation mechanism that creates multiple negative links for one node still deserves consideration, which is planned for future work.

The inference function \( g_\psi(X, A) \) is a GCN working on the original graph and the computation complexity of layer \( l \) is \( \mathcal{O}(|\mathcal{E}| K(l) K(l-1)) \) where \( |\mathcal{E}| \) is the total number of the links in the original graph [3]. For the graph with generated negative links, the total link number is \( |\mathcal{E}| + N \) as there is one negative link for each node. Thus, the computation complexity for the layer \( l \) of \( g_\psi(X, A^*) \) is \( \mathcal{O}((|\mathcal{E}| + N) K(l) K(l-1)) \) which is slightly higher than that of \( g_\psi(X, A) \) due to the increasing of the link number. The original computation cost for generating negative links is \( \mathcal{O}(N^2) \) since we need to compute \( \omega_{ij} \) for \( 1 \leq i, j \leq N \). But, in our proposed training algorithm, it is effectively lowered to \( \mathcal{O}(NT) \) by randomly sampling the potential negative neighbors. And since usually we have \( N \gg T \), the computation cost for negative link generation is significantly reduced.

The proposed method can be alternatively regarded as a data-augmentation technique since the weights of the GCN, i.e., \( \psi \), is trained based on both the graph with negative links and the original graph. In this way, the GCN is trained on more data so that the generalization performance can be effectively enhanced. From this perspective, this work shares some similarities with the adversarial training algorithms [86, 87]. In a adversarial training algorithms, besides the supervised loss of the original training data, the loss function also contains the divergence between the prediction based on the original data and the prediction based on the data with adversarial perturbations. The adversarial perturbations are added on the node features to provide data augmentation. In comparison, our method achieves the data augmentation by modifying the graph topology, i.e., adding negative links.
2.7 Experiment Results

In this section, the performance of the proposed method is empirically assessed on three widely used benchmark datasets in different settings. Moreover, some auxiliary experiments are also designed to analyze the model.

2.7.1 Experiment Settings

The performance of the proposed method are tested on three citation networks, i.e., Cora, Citeseer, and Pubmed [93]. In these datasets, nodes represent documents that are categorized into different classes; the feature of a node is a bag-of-words vector; and two nodes are connected by a bidirectional link if either one occurs in the citation list of the other. The basic statistics of these three citation networks are listed in Table 1. For each dataset, 20 labels per class are provided for training; 500 labels are used for validation, and 1000 labels are used for testing. We strictly follow the data split protocol of [94], which is commonly applied by most of the prior methods. The performance of the GCN models are evaluated based on the testing accuracy in different settings. The following comparative methods are included for the performance evaluation: GCN [3], GAT [55], Bayesian GCN [91], LSM-GAT and SBM-GCN [92], DisenGCN [90], GMNN [89], O-BVAT [86].

In all experiments, the hyperparameters of our model are kept the same. The GCN in our model consists of two layers \((L = 2)\) with 16 hidden units, i.e. \(K^{(1)} = 16\). The ReLU activation function is applied at the output of the first

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Nodes</th>
<th>Links</th>
<th>Features</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>2708</td>
<td>5429</td>
<td>1433</td>
<td>7</td>
</tr>
<tr>
<td>Citeseer</td>
<td>3327</td>
<td>4732</td>
<td>3703</td>
<td>6</td>
</tr>
<tr>
<td>Pubmed</td>
<td>19717</td>
<td>44338</td>
<td>500</td>
<td>3</td>
</tr>
</tbody>
</table>
layer. Dropout [95] is applied at the input of the two layers and the dropout rate is set to 0.5. These parameters above follow the classic settings of prior works [3, 89, 86, 91]. $f_\theta(z_i, z_j)$ is instantiated by a three-layer fully connected neural network, where the sigmoid activation function is applied at the hidden layer and there is no activation function at the output layer. The hidden units number $K^{(f)}$ for $f_\theta(z_i, z_j)$, the sampling volume $T$ for each class in Algorithm ??, and the coefficient $\alpha$ in Eq. (20) are jointly searched according to the classification performance on the validation set of Cora. The search space is specified as follows: $K^{(f)} \sim [8, 16, 32, 64]$, $D \sim [32, 64, 128, 256]$, $\alpha \sim [0.6, 0.8, 1.0, 1.2, 1.4]$. According to the average validation accuracy of 20 trials in each setting, these parameters are set to $K^{(f)} = 16$, $T = 128$, $\alpha = 0.8$. The RMSProp optimizer [96] is employed to train the model where the initial learning rate and the weight decay coefficient are set to 0.05 and 5e-4, respectively.

2.7.2 Node Classification Performance

The node classification performance of our method is evaluated by the testing accuracy with different experiment settings. With the first setting, the model is trained on the samples of the training set and the validation set is utilized to early stop the training. This setting is applied by numerous previous works and the comparison with these methods is shown in Table 2. The performance of the comparative methods are directly copied from the original literature and the results of our model are the average classification accuracies of 50 trials. In Table 2, the result with a bold marker is the best performance on that dataset. It can be seen that our method achieves the best performance on all datasets. Moreover, the performances of our model on Cora and Citeseer past t-test with the significance level of 0.001 compared with the second-best methods, which are marked by ‘⋆’ in Table 2. On Pubmed, the p-value for the t-test of our method is 0.002.
Table 2. Classification Accuracies (%) with the Standard Data Split. The **Bold Marker** Denotes the Best Performance on that Dataset, and the Star Marker (⋆) Means the Best Algorithm Pasts T-test Compared with the Second-best Algorithm with the Significance Level of 0.001. The (±) error bar denotes the standard deviation and for some of the methods, this term is omitted for not being reported by the original literature.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td>GAT</td>
<td>83.0 ± 0.7</td>
<td>72.5 ± 0.7</td>
<td>79.0 ± 0.3</td>
</tr>
<tr>
<td>Bayesian GCN</td>
<td>81.2 ± 0.8</td>
<td>72.2 ± 0.6</td>
<td>76.6 ± 0.7</td>
</tr>
<tr>
<td>LSM_GAT</td>
<td>82.9 ± 0.3</td>
<td>73.1 ± 0.5</td>
<td>77.6 ± 0.7</td>
</tr>
<tr>
<td>SBM_GCN</td>
<td>82.9 ± 0.3</td>
<td>74.0 ± 0.3</td>
<td>77.4 ± 0.4</td>
</tr>
<tr>
<td>DisenGCN</td>
<td>83.7</td>
<td>73.4</td>
<td>80.5</td>
</tr>
<tr>
<td>GMNN</td>
<td>83.7</td>
<td>73.1</td>
<td>81.8</td>
</tr>
<tr>
<td>O-BVAT</td>
<td>83.6 ± 0.5</td>
<td>74.0 ± 0.6</td>
<td>79.9 ± 0.4</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>84.4 ± 0.5</strong></td>
<td><strong>74.8 ± 0.4</strong></td>
<td><strong>82.2 ± 0.5</strong></td>
</tr>
</tbody>
</table>

The second experiment setting allows us to use the samples in the validation set for training, which is also adopted by the following two papers: [97] (GGP-X) and [98] (GIL). We compare our method with the results claimed in these two papers. Since we use the validation set to train our model, we no longer do early stopping during training. Instead, we train our model for 200 epochs in each trial and report the average results of 50 trials in Table 3. It can be observed that our method outperforms others for all three datasets and the improvements are statistically significant.

Table 3. Classification Accuracies (%) with Validation Set being Utilized for Training. The **Bold Marker** and the Star Marker (⋆) Are Consistent with the Definitions in Table 2

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGP-X</td>
<td>84.7</td>
<td>75.6</td>
<td>82.4</td>
</tr>
<tr>
<td>GIL</td>
<td>86.2</td>
<td>74.1</td>
<td>83.1</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>87.1 ± 0.4</strong></td>
<td><strong>76.5 ± 0.3</strong></td>
<td><strong>84.0 ± 0.4</strong></td>
</tr>
</tbody>
</table>

We also test our method under the data-scarce situation where the training
sample is reduced to the half, \textit{i.e.}, ten samples for each class. This experiment setting is also applied by [91] (Bayesian GCN) and [92] (LSM-GAT and SBM-GCN). In this experiment, we randomly select 10 samples for each class to create a reduced-label training set. Then 10 independent trials are conducted for this training set. We repeat this process five times, such that 50 trials in total are implemented with the reduced-label setting. The average testing accuracies are reported in Table 4. We can see that our method significantly outperforms the other two algorithms.

Table 4. Classification Accuracies (%) with Reduced-label Training. The \textbf{Bold Marker} and the Star Marker (⋆) Are Consistent with the Definitions in Table 2.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian GCN</td>
<td>76.6 ± 0.8</td>
<td>70.8 ± 0.6</td>
<td>72.3 ± 0.8</td>
</tr>
<tr>
<td>LSM-GAT</td>
<td>79.2 ± 0.4</td>
<td>69.1 ± 0.4</td>
<td>69.9 ± 0.3</td>
</tr>
<tr>
<td>SBM-GCN</td>
<td>78.0 ± 0.2</td>
<td>70.3 ± 0.6</td>
<td>71.0 ± 0.4</td>
</tr>
<tr>
<td>Ours</td>
<td>\textbf{82.5} ± 0.9</td>
<td>\textbf{71.8} ± 0.7</td>
<td>\textbf{75.0} ± 0.6</td>
</tr>
</tbody>
</table>

\subsection*{2.7.3 Model Analysis}

Now we conduct some auxiliary experiments to analyze our method. In our model, the feature propagation operation is moved from the input to the output for each layer. This modification is important for our method. To demonstrate this point, we implement our method with both the original propagation model and our modified propagation model. The average testing results of 10 independent trials on three benchmark datasets are reported in Table 5. It can be observed that the modified feature propagation model consistently improves the performance of our method over the three datasets.

In our method, the label distribution calculated by $g_\psi(X,A)$ and the label distribution calculated by $g_\psi(X,A^*)$ are regularized to be similar to each other by minimizing their cross entropy over the unlabeled nodes, which means these two
models learn interactively. If we substitute \( \hat{Y}^* \) with \( \hat{Y} \) for Eq. (18) and (19), our method will convert to a self-training algorithm [99], which annotates the unlabeled nodes with its prediction and use these annotated nodes to train itself. The comparison of our method and the self-training counterpart is shown in Table 6. We see that our method is able to generate better results than the self-training algorithm, which demonstrates the benefit of the negative links.

Table 5. Comparison of Node classification Accuracies (%) of Our Method under Different Feature Propagation Model

<table>
<thead>
<tr>
<th>Propagation model</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>( H^{(l)} = \sigma(\hat{A}^*H^{(l-1)}W^{(l)}) )</td>
<td>83.6</td>
<td>72.9</td>
<td>81.6</td>
</tr>
<tr>
<td>( H^{(l)} = \hat{A}^*\sigma(H^{(l-1)}W^{(l)}) )</td>
<td>84.4</td>
<td>74.8</td>
<td>82.2</td>
</tr>
</tbody>
</table>

The next experiment aims to inspect the learning process of the negative link generation model. For this purpose, we check the weights of the generated negative links during the training process, which is illustrated by Fig. 6. In this figure, each boxplot shows the statistic of the magnitudes of negative links generated in 10 consecutive training epochs, where the box specifies the first quartile and the third quartile; the orange bar in the box shows the median; the whiskers under and above the box extend to the minimum and the maximum; and the green triangle indicates the mean. For the Cora dataset, there are six representatives \( v_{i_{c}} \) of the potential negative neighbors for a node since the total class number is seven and we do not seek negative neighbors from the same class. If \( \omega_{ij} \) for all potential negative neighbors are similar, the magnitude of the generated negative link will be low and

Table 6. Comparison of Node Classification Accuracies (%) with Self-training

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-training</td>
<td>82.8</td>
<td>73.0</td>
<td>80.6</td>
</tr>
<tr>
<td>Ours</td>
<td>84.4</td>
<td>74.8</td>
<td>82.2</td>
</tr>
</tbody>
</table>
the minimum is 0.167. This corresponds to the situation where our negative link generation model cannot tell which potential neighbor is better than the others. From Fig. 6(a), we see that at the beginning, the weights of the generated negative links are very low while as the training goes, the generated links tend to have large weights which indicates our model learns to find the special node among other potential negative neighbors. A similar phenomenon can be found for the Citeseer dataset from Fig. 6(b). However, base on Fig. 6(c), the learning procedure is not ideal for the Pubmed dataset, which may also be the potential reason for the limited performance improvement of our method on this dataset. It is possible that learning a negative link pattern is more difficult on Pubmed since it has many more nodes and edges, which can consequently lead to more complex node relations.

Finally, we present the visualization of the node feature extracted by the first layer of our GCN model for the testing samples. The corresponding t-SNE embedding is plotted in Fig. 7. For the three datasets, the node feature extracted by our model is plotted at the right column of three subfigures with Fig. 7(a), Fig. 7(c), and Fig. 7(c) respectively corresponding to Cora, Citeseer, and Pubmed. We also provide the visualizations of the features extracted by the GCN model from [3] as a comparison, which lie at the left column in Fig. 7. In each subfigure, the nodes of different classes are represented by different colors. It can be observed that for Cora and Citeseer, our method is able to extract more discriminative features than the GCN baseline. For Pubmed, the improvement is not that obvious. This phenomenon is consistent with the observation from Fig. 6(c). Thus, improving the performance of our method on large scale datasets will be the focus of our future work.
2.8 Chapter Summary

In this chapter, a novel learning approach has been introduced, which learns to infer node labels by further considering the inverse correlated nodes. A trainable negative link generation model has been proposed to generate negative links that are beneficial for the classification performance. To avoid searching neighbors from the whole dataset, which is of high computation cost, this chapter has designed a high-efficiency training algorithm that randomly searches negative neighbors in a small subset of nodes. Currently, the proposed method conservatively generates one neighbor of each node while a generation mechanism that creates multiple negative links for one node is planned for future work. This chapter also discuss the differences and similarities between the proposed method and the prior methods. Experiments on three widespread benchmark datasets have demonstrated that the proposed method can generate better or competitive performance compared with state-of-the-art methods. Moreover, the performance improvement of the proposed method on the large scale dataset appears limited compared with the performance on the small dataset. Therefore, research on large scale datasets is considered as the future work.
Figure 6. Boxplots of the magnitude statistics of generated negative links in every 10 consecutive training epochs for three datasets: (a) Cora, (b) Citeseer, (c) Pubmed. The box specifies the first quartile and the third quartile. The orange bar in the box shows the median. The whiskers under and above the box extend to the minimum and the maximum. The green triangle indicates the mean.
Figure 7. T-SNE embedding plot of the feature extracted by the first layer of the GCNs on three datasets: (a) Cora, (b) Citeseer, (c) Pubmed. The plots lie at the left column exhibit the node features extracted by the original GCN model of [3]. The plots lie at the right column exhibit the node features extracted by our model.
CHAPTER 3

Energy Efficiency Optimization for Cooperative Spectrum Sensing

3.1 Background

Cognitive radio, as one of the most promising solutions for improving radio spectrum efficiency, has attracted tremendous research focus in recent years [18, 19, 21]. In contrast to the traditional static spectrum management policy, a cognitive radio system is able to dynamically allocate the available spectrum resource. Generally, in a cognitive radio system, spectrum bands are continuously detected and once a spectrum band is left idle, it will be assigned to the user in need. It is evident that an accurate detection procedure, also known as spectrum sensing, is essential for a cognitive radio system [11, 12, 101]. When implementing spectrum sensing with a single sensor, the performance can be critically degraded in shadowing or multipath fading environments. Accordingly, cooperative spectrum sensing is proposed to improve the sensing accuracy by exploiting the spatial diversity of multiple distributed sensors [102, 103, 13, 104, 105]. According to the communication topology, the cooperative sensing paradigms fall into three categories: centralized, relay-assisted, and distributed [13]. For the first two categories, a fusion center is necessary to collect the sensing data from the sensors and make the detection decision with a fusion rule. In this way, the fusion center can suffer from severe traffic overload. By contrast, distributed cooperative sensing accomplishes the collaboration through a local iterative consensus algorithm, where each sensor only processes the data from its neighbors [35, 36].

Regardless of which cooperative sensing paradigm is used, the signal detection devices are indispensable to collect and process the sensing data. Traditionally, the spectrum sensing module is integrated with every SU, which will increase the manufacturing cost of the wireless devices. As a result, several researchers advocate
separating this functionality from the SUs and constructing a specialized sensor network to provide the sensing service [76]. The sensor network is supposed to operate continuously to provide a real-time spectrum map for SUs. Consequently, energy consumption can be considerable over time. This issue could be more severe for a distributed cooperative sensing system since it relies on an iterative algorithm. The academic community has extensively explored the energy efficiency problem for fusion-center-based cooperative sensing [76], while to the best of our knowledge, limited research has been conducted on distributed paradigms. Moreover, due to the difference of the collaboration mechanism, the methodologies proposed for fusion-center-based cooperative sensing are not applicable to the distributed counterpart.

Under this circumstance, this chapter specifically develops an optimization framework to search the energy-efficient distributed sensing scheme. The objective is to find a minimum subset of sensors that not only has an effective topology for the distributed sensing algorithm but also satisfies the detection performance requirement. It is clear that searching the optimal solution is a combinatorial optimization problem, and the exact optimal solution of such a problem, especially for a large-scale network, is almost impossible to obtain as the state space of the solution grows exponentially with the sensor quantity. Fortunately, the recent development of artificial intelligence (AI) methodologies have brought significant breakthroughs for such complex problems in many areas. Several combinatorial optimization problems have been studied through AI-based approaches, achieving state-of-the-art performance [106, 107].

Inspired by the latest research and technology development in the community, this chapter proposes a deep learning framework to address the critical challenges for energy-efficient distributed cooperative spectrum sensing application. The pro-
posed architecture is built based on two learning techniques: GNN and RL.

### 3.2 System Model

The considered cognitive radio network in this research is depicted in Fig. 8. The considered cognitive radio network consists of PUs, SUs, and spectrum sensors.

Figure 8. A sensor-aided cognitive radio network with energy detection based distributed cooperative sensing [4]: (a) sensor-aided cognitive radio network; (b) energy detection based distributed cooperative sensing. In the energy detection illustration, the node color indicates the local SNR level of the sensor measurement. Each node recursively communicates with its neighbor to fuse these measurements with a consensus protocol until convergence. The converged result is then compared with a pre-set threshold to infer whether the target spectrum band is idle or occupied.

The sensors collaborate with each other through a consensus algorithm to detect the activities of the PUs. The detection results are reported to the spectrum policy server via the nodes nearby. Then the spectrum policy server will build a real-time spectrum map and allocate the available spectrum bands to the SUs. For the spectrum sensing purpose, sensor $i$ collects data on the target spectrum band, which is

$$x_i(k) = \begin{cases} v_i(k) & H_0 \\ h_i s(k) + v_i(k) & H_1 \end{cases}$$

(21)
In this equation, $H_0$ and $H_1$ respectively represent the hypotheses of PU signal being absent or present; $s(k)$ is the signal of the PU; $h_i$ is the channel gain from the PU transmitter to sensor $i$; and $v_i(k)$ is the zero-mean additive white Gaussian noise with variance being $\sigma_i^2$.

The sensors are spatially distributed, and each sensor can exchange information with its neighbors. The communication relationship of the sensor network can be described by a undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. $\mathcal{V} = \{v_i\}_{i=1}^N$ is the vertex set with $v_i$ representing the $i$-th sensor. $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and the edge $(v_i, v_j)$ exists only if sensor $i$ and sensor $j$ can communicate with each other. The neighboring vertices of $v_i$ compose the set $\mathcal{J}_i$ in which $(v_i, v_j)$ exists for $\forall v_j \in \mathcal{J}_i$. In this work, the sensor network is assumed to be built to a regular grid with variations. In this case, each sensor does not have too many neighbors. The variations are introduced to simulate the real-world environment since the sensor’s installation position may be affected by other constraints in reality. The cooperative sensors typically use a common detection technique, such as matched filter, cyclostationary feature detection, or energy detection, to detect the spectrum utilization status. Among these detection methods, energy detection is one of the most popular approaches due to its simplicity and independence on the specific form of the PUs signal. Therefore, this work adopts energy detection based distributed spectrum sensing algorithm [36] for this sensor network. With this algorithm, the sensor network obtain a sensing result through three steps. First, each sensor $i$ measures the test statistic over $M$ consecutive samples on the target spectrum band, which is

$$y_i = \sum_{k=0}^{M-1} |x_i(k)|^2.$$  \hspace{1cm} (22)

Then, the sensors update their measurements iteratively with the following
weighted consensus protocol [36]:

\[ y_i(t + 1) = y_i(t) + \frac{\alpha}{\delta_i} \sum_{v_j \in N_i} (y_j(t) - y_i(t)), \]  

(23)

where \( t \) is the iteration number; \( \alpha \) is the iteration step size; and \( \delta_i \) is the weighting ratio. \( \delta_i \) is designed according to the channel condition of the sensor, which will be elaborated later. It has been proved that the test statistic of each sensor will converge to a common value, which is

\[ y_c = \frac{\sum_{i=1}^{N} \delta_i y_i(0)}{\sum_{i=1}^{N} \delta_i}. \]  

(24)

Clearly, \( y_c \) is the weighted average of the initial measurements of all sensors with \( w_i = \delta_i / \sum_{i=1}^{N} \delta_i \) being the weight of the \( i \)-th sensor. Finally, \( y_c \) is delivered to the spectrum server by the neighboring sensors, and the target spectrum band will be claimed to be idle if \( y_c \) is less than a predetermined threshold \( \gamma \); otherwise, the spectrum band is deemed as occupied. This distributed cooperative sensing procedure can be illustrated by the right part of Fig. 8 (b).

It is essential for the sensor network to perform spectrum sensing continuously because the SUs must vacate the spectrum band if the PU reclams the band. As a result, the sensor network can consume tremendous energy over time. In this case, energy efficiency is an imperative consideration for the sensor network. While energy efficiency problem has been substantially investigated for fusion center based sensor networks, limited understanding and techniques are available for distributed cooperative sensor network given its unique characteristics. To this end, in the following section, we develop an optimization paradigm for the energy efficiency problem of the distributed collaborative sensor network.

### 3.3 Energy-Efficient Distributed Cooperative Spectrum Sensing

According to the central limit theorem, \( y_i \) asymptotically follows the normal distribution if \( M \) is large enough (e.g., \( m \geq 10 \) in practice). Defining the transmit-
ted signal energy of the PU and the local SNR of sensor $i$ by $E_s = \sum_{k=0}^{M-1} |s(k)|^2$ and $\eta_i = E_s |h_i|^2 / \sigma_i^2$, the mean and the variance of $y_i$ can be expressed by

$$E(y_i) = \begin{cases} M\sigma_i^2 & H_0 \\ (M + \eta_i)\sigma_i^2 & H_1 \end{cases}$$

(25)

$$\text{Var}(y_i) = \begin{cases} 2M\sigma_i^4 & H_0 \\ 2(M + 2\eta_i)\sigma_i^4 & H_1 \end{cases}$$

(26)

Since $\{y_i\}_{i=1}^N$ are independent normal random variables, $y_c$ are also normally distributed with mean

$$E(y_c) = \begin{cases} M\sigma^T w & H_0 \\ (M\sigma + E_s g)^T w & H_1 \end{cases}$$

(27)

and variance

$$\text{Var}(y_c) = \begin{cases} w^T \Sigma_{H_0} w & H_0 \\ w^T \Sigma_{H_1} w & H_1 \end{cases}$$

(28)

In (27) and (28), the quantities $\sigma$, $w$, $g$, $\Sigma_{H_0}$, and $\Sigma_{H_1}$ are defined by

$$\begin{align*}
\sigma &= [\sigma_1^2, \sigma_2^2, \ldots, \sigma_N^2]^T \\
w &= [w_1, w_2, \ldots, w_N]^T \\
g &= [|h_1|^2, |h_2|^2, \ldots, |h_N|^2]^T \\
\Sigma_{H_0} &= 2M \text{diag}^2(\sigma) \\
\Sigma_{H_1} &= 2M \text{diag}^2(\sigma) + 4E_s \text{diag}(g) \text{diag}(\sigma).
\end{align*}$$

With this $y_c$, the performance of the sensor network can be evaluated by the probabilities of false alarm ($P_{fa}$) and detection ($P_d$):

$$\begin{align*}
P_{fa} &= \Pr(y_c > \gamma | H_0) = Q \left( \frac{\gamma - E(y_c | H_0)}{\sqrt{\text{Var}(y_c | H_0)}} \right) \\
P_d &= \Pr(y_c > \gamma | H_1) = Q \left( \frac{\gamma - E(y_c | H_1)}{\sqrt{\text{Var}(y_c | H_1)}} \right)
\end{align*}$$

(29)

where $Q(\cdot)$ is the complementary cumulative distribution function of a variable that follows the normal distribution with zero mean and unit variance. The relationship
of $P_{fa}$ and $P_d$ can be further derived by

$$\begin{align*}
P_d &= Q \left( \frac{Q^{-1}(P_{fa}) \sqrt{w^T \Sigma_{H_0} w - E_s g^T w}}{\sqrt{w^T \Sigma_{H_1} w}} \right). \tag{30}
\end{align*}$$

For a certain $P_{fa}$, the optimal weights design can be found by maximizing $P_d$. However, the accurate analytic solution is of high mathematical complexity. By defining the deflection coefficient as

$$\begin{align*}
d^2(w) &= \frac{[E(y_c|H_1) - E(y_c|H_0)]^2}{\text{Var}(y_c|H_0)} = \frac{(E_s g^T w)^2}{w^T \Sigma_{H_0} w}, \tag{31}
\end{align*}$$

an alternative near optimal weights design is discovered by maximizing $d^2(w)$ [108, 36]. The rationale behind this approach is that for a given $P_{fa}$, a large $d^2(w)$ will lead to a high $P_d$ when the sensors are under a low SNR channel condition. Moreover, the low SNR channel condition is one of the most important reasons for building a cooperative sensor network. By maximizing $d^2(w)$ over $w$, the obtained near optimal weights can be concisely expressed by

$$\begin{align*}
w_{opt} &= \Sigma_{H_0}^{-1} g. \tag{32}
\end{align*}$$

The $i$-th component of $w_{opt}$, i.e., $\frac{\eta_i/\sigma_i^2}{\sum_{i=1}^N (\eta_i/\sigma_i^2)}$, is the weight of sensor $i$. Consequently, the weighting ratio of the consensus protocol in (24) can be easily derived by $\delta_i = \eta_i/\sigma_i^2$. Applying $w_{opt}$ to (31), we have

$$\begin{align*}
d^2(w) &= \frac{E_s^2}{2N} \sum_{i=1}^N \frac{h_i^4}{\sigma_i^4}. \tag{33}
\end{align*}$$

We can see that in terms of improving $d^2(w)$, the contributions from the sensors are independent, which is $h_i^4/\sigma_i^4$ for the $i$th sensor. And it is clear that a sensor under a high local SNR is able to increase $d^2(w)$ substantially while the contribution from a sensor experiencing low SNR can be negligible. Therefore, it is not reasonable to invest energy on a sensor of low SNR to pursue a slightly increase of $d^2(w)$. 41
Based on the analysis above, we define the following energy efficiency optimization problem for the distributed spectrum sensing.

According to (33), we define the performance gain for sensor \( i \) as

\[
p_i = \frac{h_i^4}{\sigma_i^2}.
\]

(34)

An effective distributed cooperative spectrum sensing scheme involves a set of sensors \( \{v_i\}_{l=1}^L \) that can not only implement the distributed spectrum sensing algorithm, but also guarantee the condition of

\[
\sum_{l=1}^{L} p_{n_l} \geq \Theta,
\]

(35)

i.e., the summation of their performance gains surpass a predetermined threshold (\( \Theta \)). From energy efficiency prospective, it is desirable to have a \( \{v_i\}_{l=1}^L \) with minimum \( L \). Based on all these discussions, it can be inferred that for a spectrum band, an energy-efficient spectrum sensing scheme is a minimum subset of connected sensors that can achieve the target spectrum sensing performance (deflection coefficient).

The threshold \( \Theta \) is determined by the target spectrum sensing performance. If a high sensing performance is requested, \( \Theta \) will be set to a large value, such that the sensor set can generate a large \( d^2(\mathbf{w}) \) and correspondingly, a high sensing performance. Moreover, the upper limit of \( \Theta \) is determined by (33), which is

\[
\overline{\Theta} = 2N d^2(\mathbf{w}_{opt})/E_s^2.
\]

This value corresponds to the situation where all the sensors are included for the cooperative spectrum sensing.

3.4 Deep Reinforcement Learning Based Solution

3.4.1 Sequential Sensor Selection

Only considering the condition of (35), the optimal energy-efficient sensor set can be found by selecting sensors one by one according to the descending order of their performance gains until \( \Theta \) is surpassed. However, the applied distributed
spectrum sensing algorithm requires the topology of the cooperative sensors being connected, which makes this optimization problem no longer straightforward. Therefore, a sequential sensor selection procedure is proposed to discover the sensor sets that can not only satisfy the condition of performance gain threshold but also guarantee the topology constrain of distributed cooperative spectrum sensing algorithm, which is shown in Fig. 9. In this process, we begin with an empty solution and add one node a time until the threshold (Θ) is reached. At the first step, we select a node that is connected to the spectrum policy server to ensure the sensing results can be delivered to it. In the following steps, the feasible selections are restricted to the nodes that have connections to the previously selected ones. The optimal solution is characterized by the minimum number of sensors, which can be obtained through the exhaustive search. However, due to the large computational complexity, the exhaustive search is not practical, especially for the large-scale sensor network. Recently, deep learning has been successfully applied to some of the combinatorial optimization problems [107]. Inspired by these prior research, in this research, a deep learning based approach for the energy-efficient distributed cooperative sensing.

![Sequential sensor selection process for distributed cooperative sensing](image-url)
3.4.2 Proposed Model

The proposed deep learning framework is obligated to make decisions for the sensor selection process presented in the earlier section. At each step, the input is the communication graph with each node carrying a two-dimensional tag, where the first component is the performance gain of the node normalized by the target threshold, and the second component denotes whether the sensor has been selected or not with ‘1’ or ‘0’. Our method estimates the Q values of all legitimate actions based on this graph-structured data. As the legitimate actions vary at different steps, designing a consistent action representation is not forthright. Considering the fact that the state transition is deterministic in the sensor selection process, we first use the future graph states to represent the corresponding actions. Then, these action representations are input to a value network separately to estimate the corresponding Q value. Finally, the selection is made with a greedy policy.

The specific procedure is depicted in the yellow box of Fig. 10. At each step, we first infer the associated future states of all the feasible actions by modifying the second component of the tags for the corresponding nodes. Then, these inferred graph-structured data are deemed as the action representations and input to the value network independently. Since this procedure is equivalent to employing multiple graph value networks with shared weights, we draw one graph value network for each action to facilitate the visualization. The graph value network estimates the action values through two steps: feature extraction and value estimation. The feature extraction process aims to embed graph-structured data to a distinctive feature utilizing a particular graph neural network, Structure2Vec [110]. Structure2Vec generates the n-dimensional feature of each node via an iterative feature propagation process. At each iteration, the node feature is updated according to the features of all the neighbors and its own tag. To make it clear, we make an
Figure 10. The proposed DRL framework [4]. The upper part depicts the decision making and training procedure. The lower part illustrates the specific structure of the graph value network, where Structure2Vec is the GNN used to generate the graph feature, and the fully connected neural network is used to calculate the action value.

Illustration with node two. First, the two-dimensional tag of a node is mapped to the feature space through the data-to-feature mapping matrix while its neighbors’ feature is mapped through the feature-to-feature mapping matrix. Then, the obtained mappings will be added together and input to a nonlinear function. The output will be applied to update the feature of node two. The n-dimensional initial feature vectors of all nodes are set to zero. At each iteration, all the node features are updated simultaneously. This process stops after T iterations, and the iteration number T is a hyperparameter that can be set to a small value based on our
experience. The obtained features are then average pooled over the whole graph to produce the representation of this graph-structured data. After that, a fully connected neural network estimates the corresponding action value according to this representation. Finally, comparing all actions, the one of the highest quality is executed.

3.4.3 Training with Reinforcement Learning

There is no doubt that the action evaluation accuracy directly affects the quality of the generated solution. Therefore, a training process is essential for the proposed deep structure to learn the correct estimation. It is clear that the problem fulfills the Markov property, which naturally leads to employ RL as the training method. RL is a powerful tool that enables an agent to learn an optimal policy for a Markov decision process through interactions with the environment [2]. In reinforcement learning, the agent receives a reward from the environment at each time step, and the learning algorithm can guide the agent to discover an optimal policy that maximizes the expected rewards accumulated over time. Consequently, when applying reinforcement learning to optimization or planning problems, it is critical to design a proper reward signal at each step that is compatible with the final goal. In this problem, since the optimal solution contains the fewest sensors, we set the reward for each action to

$$r = -1.$$  \tag{36}$$

In this way, maximizing the accumulated rewards is precisely equivalent to minimizing the number of sensors in the solution. In a traditional tabular reinforcement leaning approach, the action values at each step are stored in a table, and the learning refers to recursively updating the action values based on the related reward and the estimated future return. With neural network based RL, rather than adjusting the stored action value, the learning process aims to tune the weights of
the neural network such that it can accurately estimate the action qualities at all encountered states. Q-learning is a widely used off-policy algorithm for both tabular and neural network based approaches. In this work, the proposed model is trained via Q-learning. Since the precise action value under an optimal policy should equal the maximum expected future return, a Q-learning algorithm adapts the estimated action value or the estimation functions with the maximum action value of the forthcoming state and according, the training error is

\[ e(s,a) = (Q(s,a) - (\max_{a'} Q(s',a') + r(s,a)))^2 \]  

where \( s \) is the state, \( a \) is the implemented action, \( s' \) is the state after \( a \), and \( r(s,a) \) is the reward for action \( a \), i.e., -1. The Q-learning process is illustrated in Fig. 10. At each step, the action values are predicted via the proposed architecture. Then the \( \epsilon \)-greedy policy is applied to decide the action, where there is a probability \( \epsilon \) that the action will be randomly selected. This randomness is necessary because the action value can be incorrectly estimated at the beginning, and it is beneficial to implement explorations. With the applied action, a reward is feedback and a corresponding future state. The actions at this future state will also be evaluated by the model. Then the summation of the maximum Q value of these actions and the reward is used as the target of the predicted Q value of the executed action. Finally, to make the prediction close to the target, the gradient descent algorithm is utilized to tune the mapping matrices in Structure2Vec and the weights of the fully connected neural network. The sensor-aided cognitive radio network may be built at different areas to deal with the local spectrum shortage problem. Hence, the proposed model is expected to learn a generalized policy, such that it is able to find the energy-efficient cooperative scheme for a class of sensor networks. Similar to other reinforcement learning problems [38, 107], numerous training cases are essential for achieving this goal. Obviously, it is not possible to construct sensor
network entities for training purposes. Instead, this work assumes that the different sensor networks are the instances of a common model, and training cases can be generated with this model. The rationale behind this assumption is that cognitive radio networks are most likely to be built in the area of high user density, such as some big cities. In these areas, the environments are similar, such that it is possible to find a generalized model. In this work, since the modeling strategy is independent of the proposed method, a simplified sensor network model is used to test the effectiveness of the proposed method.

3.5 Simulation Results

This section conducts numerical simulations to verify the performance of the proposed approach. The sensor networks of different scales are considered, where the sensors are deployed in a two-dimensional grid structure with certain variations. The coordinate of each sensor is set to the form of \((x + \sigma x, y + \sigma y)\), where the integer \(x\) and \(y\) are the coordinates of the vertex in a regular quadrilateral grid; \(\sigma x\) and \(\sigma y\) are the Gaussian noise with zero mean and variance of 0.1 to simulate the variations. In each sensor network, the spectrum policy server locates in the central area. The communication range of each sensor is set to 1.35. The sensors are assume to randomly experience three levels of SNR, and accordingly, their performance gains are set to three different values: 0.1, 0.3, and 0.9. In the experiments, the thresholds are set to larger values as the networks scale up. The reason for this setting is that building a large sensor network usually aims to obtain a high sensing performance. Based on this model, 1000 training cases and 1000 testing cases are generated for each network scale. The parameters of the sensor networks are summarized in Fig. 11.

The proposed model is trained with the training cases and apply it to select the nodes for the testing cases. A good performance is indicated by a low averaged
number of the selected nodes of all test networks. To better evaluate our approach, the testing cases are also solved with a greedy heuristic (GH) based approach and a genetic algorithm (GA) [111] based approach as comparisons. The idea of the applied greedy algorithm is straightforward, which always selects the sensor with maximum performance gain at each step. The GA based approach is designed based on the priority-based encoding method [112]. In this approach, each sensor is assigned a priority value, and the priority values of all nodes compose the genetic representation of the solution. With this representation, we obtain the solution by selecting the legitimate node with the highest priority at each step. Accordingly, the fitness function is set to the accumulate rewards for this GA based approach. Fig. 12 (a) shows the performance of these three approaches. Comparing our proposed method with the GH based approach, it is clear that our proposed method always performs better for each network scale. Furthermore, we find a performance improvement increasing from 15 percent to 28 percent as the network gets larger. On the other hand, the GA based approach achieves a comparable performance. For the 16-node networks, it even performs a slightly better than our method. However, when we use these two methods to solve the test cases, there is a great difference in terms of the computation time, which is revealed in Fig. 12 (b). It should be noted that the computation time of the GH based approach is not

<table>
<thead>
<tr>
<th>Sensor network scale</th>
<th>Sensor coordinate</th>
<th>Spectrum server coordinate</th>
<th>Number of nodes under different performance gain (g)</th>
<th>Threshold (9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 3)</td>
<td>(1.5, 1.5)</td>
<td>11</td>
</tr>
<tr>
<td>24-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 4, (x, y) = (2, 2))</td>
<td>(2, 2)</td>
<td>17</td>
</tr>
<tr>
<td>36-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 5)</td>
<td>(2.5, 2.5)</td>
<td>25</td>
</tr>
<tr>
<td>48-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 6, (x, y) = (3, 3))</td>
<td>(3, 3)</td>
<td>33</td>
</tr>
<tr>
<td>64-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 7)</td>
<td>(3.5, 3.5)</td>
<td>44</td>
</tr>
<tr>
<td>80-node</td>
<td>((x + \sigma_x, y + \sigma_y)</td>
<td>0 \leq x, y \leq 8, (x, y) = (4, 4))</td>
<td>(4, 4)</td>
<td>56</td>
</tr>
</tbody>
</table>

Figure 11. Parameters of the considered sensor networks [4].
Figure 12. Performance comparison under different network scales [4]. In (a), x-axis denotes the number of nodes in the test network, and y-axis shows the average number of nodes selected by the three comparative approaches: GH (greedy heuristic) based approach, GA (genetic algorithm) based approach, and the proposed DRL (deep reinforcement learning) based approach. The performance for each network scale is based on the average of 1000 testing cases. In (b), the pie charts show the computation time (in seconds) of the GA based approach and the DRL based approach.

included because it is almost invisible compared with the other two. From Fig. 12 (b), we can observe that the time consumption of our method is obviously less than that of the GA based approach. In addition, this advantage enlarges as the network scales up.
3.6 Chapter Summary

In this chapter, we develop a RL-based approach to improve the energy efficiency of distributed cooperative spectrum sensing for sensor-aided cognitive radio network. A sequential sensor selection method has been designed to find a legitimate subset of sensors that can not only fulfill the target sensing performance, but also guarantee the topology requirement of the distributed sensing algorithm. Our model leverages a GNN and RL to estimate the action quality in the sensor selection process. Simulation results under different network scales demonstrate the effectiveness of our proposed approach. As a promising research topic, there are many interesting future research directions that can be considered. For instance, in our current study, we consider the sensor network topology to be a two-dimensional grid structure with certain variations. Therefore, it would be interesting to investigate how the proposed method can be generalized to other network topology structures.

Also, the proposed energy-efficient spectrum sensing scheme is obtained through a globalized planning style, which requires the related parameters of each sensor, such as channel gain, noise level, and signal strength, to be known. However, these parameters may be uncertain over time with the change of the surrounding environment. Therefore, distributed real-time learning will be critical for this approach to be applicable to practical communication systems in reality.
CHAPTER 4

Channel Interference Estimation for D2D networks

4.1 Background

In a D2D communication network, mobile users nearby can directly communicate with each other, which can bring about many benefits, e.g., high data rate, low power consumption, and lower delays due to the proximity of the transmitter and receiver. Moreover, D2D communication can also improve the spectrum efficiency because the low transmission power makes the spectrum spatial reuse and DSA possible [78, 113, 114]. In a D2D communication enabled network, the spectrum resources are shared by numerous D2D users, such that the D2D links have interference with each other. Therefore, interference mitigation schemes are intensively studied from different perspectives [79, 80, 81, 82]. In [79], the authors developed centralized and distributed power control algorithms for a D2D underlaid cellular system. The centralized power control aims to guarantee the coverage of the cellular users by limiting the interference created by underlaid D2D users while scheduling as many D2D links as possible; the distributed power control method maximizes the sum rate of the D2D links. In [80] and [82], pricing-based interference coordination algorithms are proposed, where the base station sets a price for each D2D resource and the D2D users update their power control strategies to minimize the interference. An energy and interference aware power control policy is designed based on a mean-field game in [81].

To achieve effective interference management [79, 80, 81, 82], it is necessary to have an accurate understanding of the interference relationship of the D2D devices. Usually, the interference information can be obtained by calculation or measurement. With the calculation-based approach, the global channel state information (CSI) is usually assumed to be available [79]. However, as the amount of CSI grows
exponentially with the number of D2D devices, the CSI can be difficult to obtain for a large-scale D2D communication network. With the measurement-based approaches \[80, 81, 82\], some of them require D2D users to implement an additional measurement process to get the interference information, which can increase the energy cost and time consumption. Moreover, some measurement process requires the transmitter to send a testing signal which and incur extra interference for the existing D2D links.

This chapter investigates the interference estimation problem for the D2D communication network. In the considered network, the D2D users share common spectrum resources and there is a base station (BS) that functions as the centralized controller to coordinate the spectrum access of the whole network. When two nearby users request to build a D2D link, they will send the necessary information to the BS station. The BS will first estimate the interference information and then send the resource allocation action, which is calculated based on the estimation, back to this D2D pair. In this work, the interference information is two-fold, which contains the interference from all other existing D2D links to the new receiver and the potential interference introduced by the new transmitter to other D2D links. The D2D network change dynamically, which means a D2D pair will terminate the D2D link when they finish the data transmission and new D2D communication requests can occur over time. This dynamic property makes the interference estimation to a challenging inference problem. In this case, this work seeks a machine learning based solution for this problem due to the great successes machine learning techniques have achieved for numerous predictions, inference, and decision-making problems of communication systems \[70, 115, 4, 116, 117, 118\]. To estimate the interference for the new-arrival D2D users efficiently, a GCN \[3\] based interference estimation framework is proposed. A training algorithm is also
designed to optimize the parameters of the proposed model.

4.2 System Model and Problem Formulation

An illustration of the considered D2D communication system is shown in Fig. 13, which is composed of a BS, D2D transmitters, and D2D receivers. The D2D transmitters and D2D receivers are randomly distributed and the BS functions as the centralized controller of all D2D pairs. There are $K$ orthogonal resource blocks (RBs) that can be used for D2D communication. There is a control channel that enables the information exchange between the BS and D2D pairs to build the D2D links. Once a D2D pair needs to communicate, they will send the request as well as the necessary information to the BS. Then, the BS broadcasts the resource allocation actions, which is calculated based on the received information, to the D2D pair to configure the D2D link. A resource allocation action is in the

![Figure 13. A D2D communication network. A transmitter directly communicate with the corresponding receiver. The D2D pairs are coordinated by the base station.](image-url)
form of power control which specifies the power allocated on each RB for a D2D transmitter. When the D2D communication is finished, the D2D pair will send a notification back to the BS.

Denoting the number of existing active D2D pairs by $N$, for the D2D communication link formed by D2D pair $i$, the received signal-to-noise-plus-interference-ratio (SINR) over the $k$-th RB is

$$
\gamma^k_i = \frac{p^k_i h^k_i}{\sum_{j \neq i}^N p^k_j h^k_{j,i} + \sigma^2}
$$

where $p^k_i$ is the power of transmitter $i$ on RB-$k$; $h^k_i$ and $h^k_{j,i}$ are the channel power gain from the $i$-th transmitter and the $j$-th transmitter to the $i$-th receiver; and $\sigma$ represents noise power. Accordingly, the achievable rate for the $i$-th D2D link over the $k$-th RB is

$$
r^k_i = B_k \log_2(1 + \gamma^k_i)
$$

where $B_k$ is the bandwidth of RB-$k$. In this work, the RBs are assumed to be equally divided and $B_k = B_0$ for $\forall k$.

Based on Eq. (38) and (39), it can be observed that there exists complex co-channel interference in the D2D network: a D2D link suffers the interference from all other transmitters and a transmitter can have interference on all other D2D links once they allocate transmission power on a common RB. For the D2D link $i$, the interference introduced by other D2D transmitters over the $k$-th RB is

$$
I^k_{\rightarrow i} = \sum_{j=1,j \neq i}^N p^k_j h^k_{j,i};
$$

and the interference that induced by transmitter $i$ is

$$
I^k_{i \rightarrow} = \sum_{j=1,j \neq i}^N p^k_i h^k_{i,j}.
$$
Since $I^k_{i\rightarrow}$ and $I^k_{i\rightarrow}$ are closely related to the data transmission rate of the D2D links, numerous resource allocation algorithms or interference management policies for D2D communications are designed based on these two quantities or their derivations [80, 81, 82]. However, $I^k_{i\rightarrow}$ and $I^k_{i\rightarrow}$ can be difficult to obtain in practice. In some prior works, $I^k_{i\rightarrow}$ are obtained through measurements, which requires the D2D pair take an extra measurement on each RB [80, 82]. This kind of approaches will make the D2D users take extra time and energy to implement the measurement. In some of the designed measurement process, the D2D transmitter need to send a test signal with a certain power [81], which can introduce extra unwanted interference. Moreover, $I^k_{i\rightarrow}$ can still be difficult to get with the measurements since we need to collect $p^k_{i\rightarrow} h^k_{i,j}$ from every D2D receiver $j$ ($j = 1, \ldots, i-1, i+1, \ldots, N$) and it is possible that a D2D receiver cannot distinguish the interference of transmitter $i$ from the interference introduced by other transmitters.

To overcome the difficulties mentioned above, this chapter investigates the interference estimation problem for the D2D communication networks. In this work, the D2D communication network is dynamically varied over time, i.e., the existing D2D users will terminate the communication when they finish their session and new D2D communication links will be built once a D2D pair generates the transmission requests. To make a clear description of the system model, we assume that the system time is slotted. At the beginning of time slot $t$, the active D2D pairs are indexed with $1, \ldots, N_t$, and there are $M_t$ new D2D pairs that are indexed from $N_t + 1$ to $N_t + M_t$. An active D2D pair refer to the D2D pair that has already started the D2D communication with the resource allocation actions specified by the BS and their power on each RB keep constant. During time slot $t$, any D2D pairs that finish the communication will send a notification to the BS and terminate the D2D transmission, such that the number of active D2D pairs is updated to $N_{t+1}$.
for the next time slot. In order to facilitate the interference management for the D2D communication network, our objective is two-fold: (1) get an estimation of the current interference level of the RBs for all new-arrival D2D pairs; (2) evaluate the potential interference of each new D2D pairs can introduce to the existing D2D links. For the first objective, we need to obtain an accurate estimation of

\[ I_{k \rightarrow i} = \sum_{j=1}^{N_t} P_j^k h_{j,i}^k \] (42)

which measures the interference from existing D2D links to a new receiver \( i \) on RB-\( k \). Note that Eq. (42) is not identical to Eq. (40) as Eq. (42) quantify the interference for a new D2D receiver while Eq. (40) is the interference for an existing D2D receiver. For the second objective, as the new D2D transmitters have not begun D2D communication, we will evaluate

\[ \overline{I}_{i \rightarrow} = \sum_{j=1}^{N_t} P_{i,j}^k h_{i,j}^k \]

\[ = \sum_{j=1}^{N_t} h_{i,j}^k \] (43)

which represents the interference that a new transmitter \( i \) can introduce to the existing receivers per unit power. Denoting the estimations for Eq. (42) and (43) by \( \hat{I}_{k \rightarrow i} \) and \( \hat{I}_{i \rightarrow} \), the estimation quality is measured by mean square error (MSE) which is calculated by

\[ E_{\rightarrow i} = \frac{1}{K \cdot M_t} \sum_{i=N_t+1}^{N_t+M_t} \sum_{k=1}^{K} (\hat{I}_{k \rightarrow i} - I_{k \rightarrow i})^2 \] (44)

\[ E_{i \rightarrow} = \frac{1}{K \cdot M_t} \sum_{i=N_t+1}^{N_t+M_t} \sum_{k=1}^{K} (\hat{I}_{i \rightarrow} - I_{i \rightarrow})^2. \] (45)

In this chapter, we propose a GCN-based estimation model to obtain \( \hat{I}_{k \rightarrow i} \) and \( \hat{I}_{i \rightarrow} \) as well as a training framework to minimize the MSE of Eq. (44) and (45), which will be presented in the next section. Note that the specific interference management policy is not the focus of this work, which will be studied in our future work.
4.3 Estimating the Interference with Graph Convolutional Network

With GCN[3] being the major technique for the proposed method, the first step is to model the D2D network as a graph. Here, a distance-based model is applied to build the graph where the transmitters and receivers are modeled by nodes and two nodes are connected by an edge if their distance is less than a threshold \( \eta \) which is a hyperparameter. Three graphs are built for a D2D network, which are \( \mathcal{G}_R = \{\mathcal{V}_R, \mathcal{E}_R\} \), \( \mathcal{G}_T = \{\mathcal{V}_T, \mathcal{E}_T\} \), and \( \mathcal{G}_{RT} = \{\mathcal{V}_{RT}, \mathcal{E}_{RT}\} \). Here, \( \mathcal{V}_x \) represents the set of nodes; \( \mathcal{E}_x \) denotes the edge set; and \( x \in \{R, T, RT\} \). \( V_R \) is composed of all receivers and an edge in \( \mathcal{E}_R \) connects two receivers that are closer than \( \eta \). \( V_T \) is composed of all transmitters and an edge in \( \mathcal{E}_T \) connects the nearby transmitters. \( V_{RT} \) contains both transmitters and receivers, and an edge in \( \mathcal{E}_{RT} \) connects a transmitter and a receiver whose distance is less than \( \eta \). These three graphs can be combined to one graph, however, for the convenience of describing the method, they are still separate at this stage, and will be combined when necessary. For these three graphs, there are three adjacency matrices associated with them, which are \( A^R \in \mathbb{R}^{(N_t+M_t) \times (N_t+M_t)} \), \( A^T \in \mathbb{R}^{(N_t+M_t) \times (N_t+M_t)} \) and \( A^{RT} \in \mathbb{R}^{2(N_t+M_t) \times 2(N_t+M_t)} \). The entries in these three adjacency matrices represent the weights of the corresponding edges in each graph; and any entry in these three matrices is set to 0 if the corresponding nodes are not connected. It should be noted that since \( \mathcal{G}_{RT} \) contains both transmitters and receivers, the node number in \( \mathcal{G}_{RT} \) is twice of the node number in \( \mathcal{G}_R \) and \( \mathcal{G}_T \). Accordingly, the dimension of \( A^{RT} \) is \( 2(N_t+M_t) \times 2(N_t+M_t) \) and the first \( N_t+M_t \) nodes correspond is assumed to be transmitters and the remaining nodes be the receivers. The specific value of the adjacency matrices is obtained through our algorithm which will be introduced later.

When estimating \( \mathcal{I}_{k \rightarrow i}^e \), \( \mathcal{G}_R \) and \( \mathcal{G}_{RT} \) are combined first. An illustration of the
obtained graph is shown in Fig. 14. In Fig. 14, the triangles represent transmitters; the circles are the receivers; the dash lines are the edges from $G_{RT}$; the solid lines are the edges from $G_R$. For a receiver node $i$, the estimation $\hat{I}_{i \rightarrow j}$ is calculated through the graph convolution process:

\begin{align*}
\text{for } i = 1, \ldots, N_t + M_t \\
&i' = i + N_t + M_t \\
&a_{ij}^{RT} = \exp^{-\alpha d_{ij}^{RT}}, j \in \mathcal{N}_i^{RT}, \\
&h_i = \sum_{j \in \mathcal{N}_i^{RT}} [a_{ij}^{RT} \sigma(1\Theta)], \\
&a_{ij}^{T} = \exp^{-\beta d_{ij}^{RT}}, j \in \mathcal{N}_i^{T}, \\
&\hat{x}_{i \rightarrow j}^k = \frac{1}{1 + \sum_{j \in \mathcal{N}_i^{RT}} a_{ij}^{T} (h_i \Psi + \sum_{j \in \mathcal{N}_i^{T}} a_{ij}^{T} h_j \Psi)}.
\end{align*}

Eq. (46) obtain the index of receiver $i$ in $A^{RT}$ since the receivers are located in the second half of $A^{RT}$. In Eq. (47), $a_{ij}^{RT}$ is the entry of $A^{RT}$ lying on row $i$ column $j$ which is the weight for the edge connecting the $i$-th receiver and $j$-th transmitter,
\( \mathcal{N}^\text{RT}_i \) is the set of transmitter neighbors of receiver \( i \) in \( G^\text{RT} \), \( d^\text{RT}_{ij} \) is the distance between the receiver \( i \) and transmitter \( j \), \( \alpha \) is a trainable parameter. Eq. (48) calculates a \( l \)-dimensional hidden variable \( h_i \) for each receiver \( i \), where \( p^k_j \) is the transmission power of transmitter \( j \) over RB-\( k \), \( \theta \in \mathbb{R}^{1 \times l} \) is a trainable weight, and \( \sigma \) is sigmoid function. In Eq. (49), \( a^R_{ij} \) is the entry of \( A^R \) lying on row \( i \) column \( j \) which is weights for the edge connecting receiver \( i \) and receiver \( j \), \( \mathcal{N}^\text{R}_i \) is the set of neighbors for receiver \( i \) in \( G^\text{R} \), \( d^\text{R}_{ij} \) is the distance between receiver \( i \) and receiver \( j \), \( \beta \) is a trainable parameter. In Eq. (50), \( \Psi \in \mathbb{R}^{l \times 1} \) is the trainable weight.

Although there is a for loop in this method, all the computation in the for loop are independent in terms of \( i \) and can be efficiently processed in parallel.

The overall process of Eq. (46) - (50), follows the general framework proposed by [3]. However, the adjacency matrix design is different. In the GCN of [3], the adjacency matrix is constant and determined by graph topology. In this work, the value of each element in the adjacency matrix is obtained through training and closely related to the distance of the nodes.

For the calculation of \( \hat{I}^k_{i \rightarrow} \), \( G_T \) and \( G^\text{RT} \) are combined, and an illustration is shown in Fig. 15. For the D2D link \( i \), \( \mathcal{I}^\text{D}_i \) is calculated through a process that is

![Figure 15. An illustration of the combination of \( G_T \) and \( G^\text{RT} \).](image-url)
similar to Eq. (46) - (50):

for \( i = 1, \ldots, N_i + M_i \)

\[
a_{ij}^{RT} = \exp^{-\alpha d_{ij}^{RT}}, j \in \mathcal{N}_i^{RT},
\]

(51)

\[
h_i = \sum_{j \in \mathcal{N}_i^{RT}} [a_{ij}^{RT} \sigma(1\Theta)],
\]

(52)

\[
a_{ij}^T = \exp^{-\beta d_{ij}^T}, j \in \mathcal{N}_i^T,
\]

(53)

\[
\hat{I}_{i \rightarrow k} = \frac{1}{1 + \sum_{j \in \mathcal{N}_i^T} a_{ij}^T (h_i \Psi + \sum_{j \in \mathcal{N}_i^T} a_{ij}^T h_j \Psi)}.
\]

(54)

In Eq. (51)-(54), \( \mathcal{N}_i^T \) is the set of neighbors for transmitter \( i \) in \( \mathcal{G}_T \), \( a_{ij}^T \) is the entry of \( A^T \) lying on row \( i \) column \( j \), which is weights for the edge connecting transmitter \( i \) and transmitter \( j \), other notations keep consistent with Eq. (47)-(50). This process first calculates hidden variable \( h_i \) for transmitter \( i \) to implicitly approximate the interference effect of transmitter \( i \) on all other receivers with Eq. (52). In Eq. (52), the numerical value ‘1’ before \( \theta \) represents a unit power. Then, the interference estimation, \( \hat{I}_{i \rightarrow k} \), is obtained through the graph convolution operation of Eq. (54).

In this model, there are four trainable parameters: \( \alpha, \beta, \Theta, \) and \( \Psi \). To make the estimation of the model accurate, a gradient descent based training algorithm is designed to optimize these parameters.

4.4 Model Training

To provide the optimization references for the proposed model, the assumption is made that when a D2D pair complete the communication, they will send the interference on each RB at the time slot when they begin the D2D communication back to the BS; and there is only a small part of the D2D users finishing communication at each time slot, such that the communication overhead on the control channel is not high. The interference information send back by D2D pair \( i \)
is denoted by $\hat{I}_{k_i}$. Note that $\hat{I}_{k_i} \rightarrow i$ may not be the interference in the current time slot because the communication of a D2D pair can continue for more than one time slot; this interference information can be obtained during the communication so that the D2D user do not need to spend additional time and energy. Since the power of each D2D transmitter is allocated by the BS, the power information of D2D pair $i$ is also available to the BS. Thus, the BS can tune the parameters of the GCN model with a gradient descent algorithm. Without loss of generality, the first $D_t$ is supposed to be activate D2D pairs terminate communication during time slot $t$. The training loss at time slot $t$ is

$$e_t = \frac{1}{K \cdot D_t} \sum_{k=1}^{K} \sum_{i=1}^{D_t} (\hat{I}_{k_i} \rightarrow i - \hat{I}_{k_i} \rightarrow i)^2.$$  \hspace{1cm} (55)

With this loss, the parameter of the model can be tuned by the gradient descent algorithm. Suppose the total time slots of the D2D network is $T$, the overall training procedure is summarized in Algorithm 2.

**Algorithm 2** Training procedure

**Parameter:** $\alpha, \beta, \Theta, \Psi$

1. for $t = 1, 2, \ldots, T$ do
2. Build $G_R, G_T, G_{RT}$
3. for $k = 1, 2, \ldots, K$ do
4. for $i = N_t + 1, N_t + 2, \ldots, N_t + M_t$ do
5. Calculate $\hat{I}_{k_i} \rightarrow i$ with Eq. (46) - (50)
6. Calculate $\hat{I}_{k_i} \rightarrow i$ with Eq. (51) - (52)
7. end for
8. Calculate the resource allocation actions with the predefined policy based on $\hat{I}_{k_i} \rightarrow i$ and $\hat{I}_{k_i} \rightarrow$ for each new D2D pairs
9. Collect $\hat{I}_{k_i}$ for the D2D pairs that complete communication
10. Calculate $e_t$ with Eq. (55)
11. Update $\alpha, \beta, \Theta, \Psi$ with $(\frac{\partial e_t}{\partial \alpha}, \frac{\partial e_t}{\partial \beta}, \frac{\partial e_t}{\partial \Theta}, \frac{\partial e_t}{\partial \Psi})$ through gradient descent
12. end for
13. end for

The reference interference for $\hat{I}_{k_i} \rightarrow$ is hard to obtain as it requires all receivers to measure the interference from transmitter $i$ and send the measurements to
the BS. Thus, in Algorithm 1, we do not train our model for $\hat{T}_{i \rightarrow}^k$. However, we find that our model can also provide accurate estimations for $\bar{T}_{i \rightarrow}^k$, which will be shown in the simulations. The potential reason for this phenomenon can be summarized as follows: (1) $T_{i \rightarrow}^k$ and $\bar{T}_{i \rightarrow}^k$ are both decided by the transmission powers and the channel power gains; (2) the transmission power in each of them are known to our model; (3) the channel power gains are decided by the locations of the transmitters and receivers as well as the environment around them; (4) the optimization reference for $\hat{T}_{i \rightarrow}^k$ is generated by the D2D users of different locations, such that the proposed model can implicitly learns a generalized feature for the power channel gains.

4.5 Simulations and Analysis

This section conducts simulations to evaluate the effectiveness of the proposed method. The D2D users are randomly distributed in a $500m \times 500m$ square area. The channel is modeled by the path loss model and independent Rayleigh fading. The proposed method is applied on two networks of different scales, where one of them has 50 D2D pairs initially and the other contains 100 D2D pairs at the beginning. The total time slot for these two systems is set to 100. At each time slot, the number of new D2D users are randomly sampled from 1 to 10, and the number of the D2D pairs that terminate communication is randomly sampled from 0 to 10. Other system parameters are listed in Tables 7, which are designed based on the experiments from [80, 82]. As the power allocation policy is independent of our method, we apply random power allocation in our simulation for simplicity, where a D2D transmitter will randomly allocate its transmission over all RBs. Note that our method can be extended to other power allocation policy without modification. An illustration of a network that contains 50 D2D pairs are shown in Fig. 16. In this figure, the circles represent transmitters, the stars are the
Table 7. System parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial number of D2D pairs</td>
<td>50, 100</td>
</tr>
<tr>
<td>Number of new D2D pairs at each time slot</td>
<td>$\mathcal{U}(0, 10)$</td>
</tr>
<tr>
<td>Number of terminated D2D pairs at each time slots</td>
<td>$\mathcal{U}(0, 10)$</td>
</tr>
<tr>
<td>Number of system RBs (K)</td>
<td>6</td>
</tr>
<tr>
<td>RB bandwidth ($B_0$)</td>
<td>180 KHz</td>
</tr>
<tr>
<td>Transmit power</td>
<td>20 dBm</td>
</tr>
<tr>
<td>AWGN noise power</td>
<td>-174 dBm/Hz</td>
</tr>
<tr>
<td>Path loss exponent</td>
<td>2</td>
</tr>
<tr>
<td>Rayleigh fading</td>
<td>$\mathcal{CN}(0, 1)$</td>
</tr>
<tr>
<td>Radius of D2D pairs (m)</td>
<td>$\mathcal{U}(10, 30)$</td>
</tr>
</tbody>
</table>

receivers, and the paired transmitter and receiver are connected by a solid line.

Figure 16. A D2D network containing 50 D2D pairs. The circles represent transmitters and the stars are the receivers. The paired D2D transmitter and receiver are connected by a solid line.

For the parameters of our method, the threshold for the distance-based graph model is set to 250m, $\Theta \in \mathbb{R}^{1 \times 10}$, $\Psi \in \mathbb{R}^{10 \times 1}$, the optimizer employed to implement gradient descent is Adam [119].

The proposed method is first implemented on the network contains 50 initial D2D users. To make a concrete evaluation of our method, we randomly initialize this network for 20 times, and the MSE is plotted in Fig. 17. In Fig. 17 (a),
the blue line is the average MSE of Eq. (44) over the 10 trails of our method, and the original line is the performance achieved by the average estimation, where \( T_{k \rightarrow i} \) is estimated by the corresponding average value of the terminated D2D pairs at each time slot. This average estimation is used as comparison since in the system, only the terminated D2D pairs send the interference back to the BS. It can observed that the estimation error continuously decreases during the training and achieves lower estimation error compared to the average estimation strategy.

Fig. 17 (b) plots the MSE of the estimation of \( \hat{T}_{k \rightarrow i} \). It can be observed that the estimation error of \( \hat{T}_{k \rightarrow i} \) constantly reduce although our model is not trained \( T_{k \rightarrow i} \), which demonstrate the claims in the previous section. Moreover, it can be seen that the final estimation error of \( \hat{T}_{k \rightarrow i} \) is larger than that of \( \hat{T}_{k \rightarrow i} \) since our model is only tuned by \( \hat{T}_{k \rightarrow i} \). It should be noted that \( \hat{T}_{k \rightarrow i} \) can not be estimated by average estimation strategy since the D2D pairs cannot distinguish the interference from a particular transmitter form other interference source.

The proposed method is also implemented on the network with 100 initial D2D pairs. The simulation results are shown in Fig 18. It can be seen that the evolution of estimation error is similar to that of Fig. 17, which demonstrate the application potential of our method on large scale networks.
Figure 18. Estimation error during the training process of the network containing 100 initial D2D pairs: (a) average MSE of $\tilde{I}_{k \rightarrow i}^k$ achieved by our method and the average estimation in 10 trials; (b) MSE of $\tilde{I}_{i \rightarrow i}^k$ achieved by our method in 10 trials where each line represent the result of one trial.

4.6 Chapter Summary

In this chapter, we study the interference estimation for D2D communication networks. The objective is to obtain accurate estimations for the interference a D2D receiver suffers on each spectrum resource and for the interference a D2D link introduces to other D2D users. A GCN-based model is proposed which provides the interference estimation based on the locations of the transmitter and the receiver for a D2D pair. The training method for the proposed model only requires the terminated D2D users to send back the interference information during its D2D communication. Simulation results demonstrate that our model can obtain the estimation of interference with low error for the D2D networks of different scales.
In a femtocell network, several APs are spatially deployed, and the wireless users can access the wireless network via the AP nearby. In this way, the distance between the transmitter and the receiver can be greatly reduced, which will bring multiple benefits, such as more reliable communications, higher throughputs, power and battery savings, and so forth [6, 27, 83]

A distinguish property of the femtocell network is that the transmitting power of an AP is low, such that it is possible to boost the spectrum efficiency via spectrum spatial reuse [84]. Moreover, the traffic loads of the femtocells can be varied both temporally and spatially [85]. Consequently, DSA [10] for femtocell network attracts the interests of numerous researchers in the community. Substantial DSA methods have been proposed to improve the spectrum efficiency from different aspects [32, 33, 28, 34, 31]. In [33], the considered channel selection problem is addressed by a distributed learning algorithm. In this algorithm, the users update their channel selection based on their own local observation, and no information exchange procedure is required. In [74], an iterative spatial adaptive play algorithm is proposed to solve the local channel selection game. With this method, one randomly picked user updates its strategy at each iteration. Meanwhile, a concurrent spatial adaptive play algorithm is designed to achieve similar performance with fewer iteration steps. These algorithms have been extended to the femtocell network, where each AP selects several channels based on its traffic loads [34]. Most of the previous approaches generate the DSA solution in an iterative searching style. However, as the network scales up, the computation time may increase a lot. Furthermore, as the spectrum demands of the femtocells can vary dynami-
cally over time, the iterative searching procedures of these approaches need to be reimplemented for each newly encountered traffic states. Therefore, for a femtocell network with dynamic traffic loads, most of the previous approaches are inefficient seeing that similar problems are solved over and over again. Accordingly, it is desirable to have a generalized spectrum access policy that can directly reveal the solution for each specific traffic state. Manually designing such a spectrum access policy, especially for a large scale femtocell network, requires solid background knowledge and problem-specific researches. Machine learning techniques have shown great potentials on solving complex problems for large-scale modern communication networks [67, 69, 71, 120]. In this case, this chapter aims to develop an autonomous machine learning based approach to discover a spectrum access policy that directly deduces the traffic loads of the network to the channel access scheme. The DSA of the femtocell network concerns which channel to access for each femtocell under the incumbent traffic load, such that it can be naturally modeled as a decision-making problem. Recent breakthroughs in machine learning have achieved tremendous successes for various decision-making problem in many other disciplines [107, 106, 121], which inspires this work.

In this chapter, the investigated femtocell network experiences different channel demands at different cells. Each femtocell is served by one AP and the corresponding AP selects a certain amount of channels from a set of available channels based on its traffic load. If two APs access a common channel simultaneously, they can interfere with each other. The interference relationship of the whole femtocell network is modeled by a distance-based conflict graph. A channel access graphical game is employed to coordinate the interference among the neighboring femtocells. In the game, each AP aims to select the channels that can maximize its expected throughput. A reinforcement learning framework is designed for solving this game.
This chapter also proposes a graph neural network based deep learning architecture to implement the proposed reinforcement learning. In this way, the learned policy is embedded in the deep architecture and possesses excellent generalization capability, which can be directly applied to the each new encounter traffic state. Moreover, the trained model can make decision for each AP in a distributed style since it generates channel access schemes based on the local information.

5.1 System Model

In this work, the considered femtocell network containing $N$ femtocells with each cell served by one AP. There are total $M$ wireless channels of the same transmission rate, which compose the set $C = \{C_1, \ldots, C_M\}$. The $n$-the AP selects $D_n$ ($D_n \leq M$) channels from $C$ to transmit data for its users. $D_n$ is jointly determined by the user number and their traffic demands$[34]$. As the user population and the corresponding traffic loads are different in each femtocell, $D_n$ varies for different APs. Within the femtocell network, if multiple APs transmit data on the same channel simultaneously, interference can be incurred. Moreover, as the femtocells are spatially distributed and the transmission power of each AP is limited, the interference only occurs among the nearby APs. To model the interference relationship of the femtocells, conflict graph is applied. The conflict graph is widely employed as a modeling tool for the interference relationship of wireless devices.

In [42], a spectrum sharing graphical game is formulated, where the network topology is assumed to be in the form of a regular lattice and the spectrum qualities differentiate among the users. Xu et al. build the interference graph for wireless users based on their location profile and propose a spatial channel selection game to achieve spectrum spatial reuse $[33]$. In [74] and [122], the authors focus on the MAC-layer interference and generate the conflict graphs for the cognitive radio system according to a distance-based model. Similar modeling strategies for fem-
Femtocell networks can be found in [123, 124, 30, 125, 43, 34]. Following the previous works [122, 34], a distance based model is employed to characterize the interference relationship of the APs, where if the distance between two APs is less than a threshold $d_0$, they can interfere with each other. Fig. 19 illustrates an example of the considered femtocell network, which contains six femtocells with a different number of users in each cell.

According to this interference model, a conflict graph $G = \{V, E\}$ can be obtained, where $V = \{1, 2, \ldots, N\}$ is the vertex set, and $E \subseteq V \times V$ is the edge set. In the graph, each vertex represents an AP, and an edge between two vertices indicates that the corresponding APs can interfere with each other. In a conflict graph, each vertex $n$ can have several one-hop neighbors, and the collection of all its neighbors are denoted by $J_n$. When multiple neighboring APs access one channel simultaneously, a transmitting coordination mechanism is necessary to avoid collisions. Following the system settings of [122, 126], CSMA is applied here.

Figure 19. An example of the femtocell network [5]. Each femtocell serves several users. The number in each femtocell is its index. Two APs can interfere with each other if their distance is less than the threshold.
In such a femtocell network, the throughput of each cell depends not only on its own channel selection but also on the actions of all its neighbors. Therefore, it is essential to design a channel access policy to improve transmission efficiency. Game theory has been widely utilized to obtain efficient channel access policies for communication networks [68, 127, 34, 69]. In this work, a graphical game is adopted to coordinate the channel access of the APs.

5.2 Graphical Channel Access Game

Denote the set of the channels accessed by AP \( n \) by \( a_n = \{C_{n1}, \ldots, C_{nP_n}\} \). With CSMA, an AP can successfully use one channel only if all other neighboring APs on the same channel keep silent. Following the definition of [122], the MAC layer interference on \( C_{ni} \) is

\[
I_{ni} = \sum_{k \in J_n} \sum_{j=1}^{D_k} \delta(C_{ni}, C_{kj}),
\]

where \( \delta(C_{ni}, C_{kj}) = 1 \) if \( C_{ni} = C_{kj} \); else \( \delta(C_{ni}, C_{kj}) = 0 \). For AP \( n \), the probability of successfully accessing \( C_{ni} \) is \( 1/(1+I_{ni}) \); and the expected achievable transmission rate on \( C_{ni} \) can be estimated by

\[
E[r_{ni}] = \frac{S_{Ci}}{1 + I_{ni}}.
\]

The quantity \( S_{Ci} \) is the transmission rate of \( C_{ni} \), which is assume to be identical for all channels. Accordingly, the expected throughput of AP \( n \) can be calculated as

\[
R_n = \sum_{i=1}^{D_n} E[r_{ni}].
\]

The graphical channel access game is denoted by \( \mathcal{F} = (\mathcal{G}, \mathcal{A}, \mathcal{U}) \), where \( \mathcal{G} \) is the conflict graph defined in the previous subsection; \( \mathcal{A} = \mathcal{A}_1 \otimes \mathcal{A}_2 \otimes \ldots \otimes \mathcal{A}_N \); \( \otimes \) represents Cartesian product; \( \mathcal{U} = \{u_n\}_{n=1}^{N} \). For vertex \( n \), \( \mathcal{A}_n \) is the set of
all feasible actions, and the quantity $u_n$ is the utility function, which is naturally defined as $u_n \equiv R_n$. Under this definition, an AP can achieve the maximum expected throughput by maximizing its own utility function. Consequently, the optimal policy can be written as $a^*_n = \arg \max_{a_n} \{u_n(a_n, a_J)\}$, where $a_J$ is the action profile of all its neighbors.

This problem and many of its variants have been studied in [42, 34, 68, 74]. Most of the proposed methods generate solutions through iterative searching processes. However, these approaches can be inefficient since the involved iterative algorithms need to be reimplemented for every encountered traffic load status. In the remaining part of this chapter, a GNN-based multiagent Q-learning algorithm is proposed, which learns a generalized policy that directly gives out the actions based on the information of the local traffic loads.

5.3 Dynamic Channel Access via Reinforcement Learning

From Eq. (57), it can be clearly observed that the expected throughput of each AP depends on the MAC layer interference of its selected channels. Therefore, a low interference level can be naturally deemed as an indication of a high channel quality. For AP $n$, define the quality of the channels as

$$Q_n = [q_{n,1}, q_{n,2}, \ldots, q_{n,M}]^T, \quad (59)$$

where $q_{n,m}$ represents the quality of $C_m$, and $q_{n,m} = -\sum_{k \in J_n} \sum_{j=1}^{D_n} \delta(C_m, C_{kj})$. Based on this definition, a reasonable policy $\pi_n$ should select $D_n$ channels of the highest quality, which can be written as

$$a_{n}^{\pi_n} = \{\{C_{nm}\}_{m=1}^{D_n} | q_{n,m} \geq q_{n,j}, \forall C_j \notin \{C_{nm}\}\}. \quad (60)$$

However, estimating the accurate value of $Q_n$ can be difficult since this quantity is jointly decided by the actions of all neighboring APs. Thus, we design a multiagent Q learning algorithm to improve the estimation accuracy and to get a near optimal
action for each AP. In our algorithm, the first step is to initialize the Q value for all APs with random value. Then, all APs will select channels according to their own Q value with an $\epsilon$-greedy policy, which means a random selection will be made with the probability of $\epsilon$. $\epsilon$ controls the exploration rate and can be gradually decreased during the training process. After the channel selection being implemented, the APs will exchange the information of their action among the neighbors and get the channel interference levels based on the received information. Then, Q value is updated by

$$q_{n,m}^{t+1} = \alpha r_{n,m} + (1 - \alpha)q_{n,m}^t$$

for all channels. In (61), $t$ is the iteration number; $\alpha$ is the learning rate; $r_{n,m}$ is the reward and is defined as

$$r_{n,m} = -\sum_{k \in \mathcal{J}_n} \sum_{j=1}^{D_k} \delta(C_m, C_{k_j}).$$

(62)

We can see that the reward of (62) shows how many neighboring APs access $C_m$. This multiagent Q learning procedure is summarized in Table 8.

This Q learning algorithm can be easily implemented via a Q table. However, as the system state grows exponentially with the number of AP, the table size can be extremely large when femtocell network scales up. Moreover, the generalization capability is quite limited since the table will not contain the Q value of a state that is not encountered in the training process. To overcome these disadvantages, a GNN-based implementation scheme next.

5.4 Graph Neural Network based Implementation

When implementing a Q learning algorithm with a neural network, the neural network functions as a mapping from state representations to the Q values of feasible actions. Traditionally, the state representation is extracted from the raw input through a deep neural network. A well-known paradigm is the deep
Initialization:

1. Set the iteration index $t = 0$;

2. Randomly initialize $Q_n^0$, for $\forall n \in \{1, 2, \ldots, N\}$.

for $t \leq T$ do:

1. For $\forall n \in \{1, 2, \ldots, N\}$, generate $a_n$ with $\epsilon$-greedy policy according to the current $Q_n^t$.

2. Implement the actions and then based on Eq.(2), calculate the corresponding \(\{I_{n1}, \ldots, I_{nM}\}\), for $\forall n \in N$;

3. For $\forall n \in \{1, 2, \ldots, N\}$ and $\forall m \in \{1, 2, \ldots, M\}$, update the estimation of the channel interference level with

\[
q_{n,m}^{t+1} = \alpha r_{n,m} + (1 - \alpha)q_{n,m}^t
\]

4. $t = t + 1$

Q network (DQN) which employs a deep convolutional neural network (CNN) to embed the raw images to a state vector [38]. However, in the graphical channel access game studied in this research, the data is graph-structured with each node carrying the local channel demand, such that a traditional CNN is no longer applicable for the feature extraction. Fortunately, graph neural networks have been proved to be powerful tools for processing graph-structured data. Consequently, we design the following graph neural network based architecture to implement the proposed multiagent Q learning algorithm, which is illustrated in Fig. 20. This architecture takes the graph data as input, where each vertex value is defined by the corresponding channel demand. A graph neural network is employed to extract the feature for each vertex. In this research, we apply a widely used graph neutral network, Graph convolutional network (GCN) [3], for this part. The GCN generates the node feature through an iterative feature aggregation procedure. At each
Figure 20. The proposed graph neural network based architecture for the estimation of channel quality [5]. In this architecture, the first part is a graph neural network, which is used to extract a feature embedding for each vertex. GCN is applied for this part which generate the vertex feature through a local information aggregation process (also known as graph convolution). Then, based on the obtained feature, each node will evaluate the channel quality via a fully connected multi-layer neural network.

iteration, the node feature is updated according to the features of all neighbors and its own data. Denoting \( Z = [z_1^T, z_2^T, \ldots, z_N^T]^T \), where \( z_n \) is the feature of the \( n \)-th vertex, the feature aggregation procedure at the \( k \)-th iteration can be written as

\[
Z^k = f_g(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} Z^{k-1} \Theta^k),
\]  

(63)

where \( f_g \) is a nonlinear activation function; \( \Theta^k \) is a trainable mapping matrix; \( \tilde{A} = I + A \) with \( A \) and \( I \) being the adjacency matrix and identity matrix respectively; \( \tilde{D} \) is the diagonal node degree matrix of \( \tilde{A} \). Note that, for a conflict graph, with \( A_{i,j} \) being the element of \( A \) laying on the \( i \)-th row and \( j \)-th column, \( A_{i,j} = 1 \) if vertex \( j \) is the neighbor of vertex \( i \). Moreover, with \( \tilde{D}_{ii} \) being the \( i \)-th diagonal element of \( \tilde{D} \), \( \tilde{D}_{ii} = 1 + \sum_{j=1}^{N} A_{i,j} \).

It can be seen that the feature update of a vertex only related to the features of its neighbors. Therefore, decomposing (63) by row, the local update equation
of vertex \( n \) can be written as

\[
z_n^k = f_g(d''_{nn}(\sum_{j \in N_n} d'_{jj}z_j^{k-1}\Theta^k + d''_{nn}z_n^{k-1}\Theta^k))
\]  

(64)

with \( d''_{ii} = 1/\sqrt{\tilde{D}_{ii}} \), which can be regarded as a normalization coefficient. It should be noted that the initial feature of a vertex is set to the corresponding channel demands, \( i.e., z_n^0 = D_n \). To make this process intelligible, we make an illustration on vertex three in Fig. 20. First, the channel demands of vertex three and all its neighbors are mapped to the feature space through \( \Theta^1 \). Then, the obtained data are normalized via the corresponding coefficients and added together. After that, the summation will be normalized by \( d''_{33} \) and input to the nonlinear function \( f_g \). Finally, the output data is used to update the feature of vertex three. This process stops after \( K \) iterations, and the iteration number \( K \) is a hyperparameter that can be set to a small value based on our experience.

Based on the obtained feature of each vertex, the corresponding Q value will be estimated through a local fully connected neural network. In this work, three-layer neural networks are applied for this part, and the Q value of the vertex \( n \) can be calculated by

\[
Q_n = W_{n,2}f_q(W_{n,1}z_n)
\]  

(65)

where \( W_{n,1} \) is the weight matrix connecting the input layer and the hidden layer; \( W_{n,2} \) is the weight matrix connecting the hidden layer and the output layer.

Since the Q value is directly related to the weights of both GCN and the fully connected neural networks, to generate an accurate estimation, we train this proposed architecture through the multiagent Q learning algorithm of Table 8. First, we randomly initialize weights, and for a specific channel demand state \( \{D_n\}_{n=1}^N \), \( Q_n \) is calculated according to Eq. (65). Then, the actions of all vertices are generated based on the \( \epsilon \)-greedy policy. With these actions, the training error
is calculated by

\[
 e = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{M} \sum_{m=1}^{M} (q_{n,m} - y_{n,m})^2
\]

(66)

Finally, to minimize this training error, all the weights are updated by the gradient descent algorithm. The whole training process is summarized in Table 9. In Table

<table>
<thead>
<tr>
<th>Initialization:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set the iteration index ( t = 0 );</td>
</tr>
<tr>
<td>2. Random initialize ( {\Theta^{k,t}}<em>{k=1}^{K}, {W</em>{t,1}}<em>{i=1}^{N}, {W</em>{t,2}}_{i=1}^{N} ).</td>
</tr>
<tr>
<td>for ( t \leq T ) do:</td>
</tr>
<tr>
<td>1. Calculate ( Q_{n}^{t} ) for ( \forall n \in {1, 2, \ldots, N} );</td>
</tr>
<tr>
<td>2. With ( \epsilon )-greedy policy, generate ( a_{n} ) based on ( Q_{n}^{t} ), for ( \forall n \in {1, 2, \ldots, N} );</td>
</tr>
<tr>
<td>3. Implement the actions and calculate the rewards based on (11);</td>
</tr>
<tr>
<td>4. For ( \forall n \in {1, 2, \ldots, N} ), calculate the training error ( e ) and update the parameters with gradient descent algorithm:</td>
</tr>
</tbody>
</table>
| \[
\Theta^{k,t+1} = \Theta^{k,t} - \gamma \frac{\partial e}{\partial \Theta^{k,t}}, 1 \leq k \leq K
\]
| \[
W_{n,1}^{t+1} = W_{n,1}^{t} - \gamma \frac{\partial e}{\partial W_{n,1}^{t}}
\]
| \[
W_{n,2}^{t+1} = W_{n,2}^{t} - \gamma \frac{\partial e}{\partial W_{n,2}^{t}}
\]
| 4. \( t = t + 1 \) |

\( \gamma \) is the learning rate of the gradient decent algorithm, and \( \Theta^{k,t}, W_{n,1}^{t}, \) and \( W_{n,2}^{t} \) denote the corresponding weights at iteration \( t \).

Since the proposed model should be applied to different traffic states, it should be trained on sufficient training cases to improve its generalization capability. Note
that since the training error of the graph neural network depends on the estimation error of all nodes, global information and centralized training are necessary. However, after training, each AP only needs to store the trained weights and generate the channel quality estimation based on the local information.

5.5 Simulation Studies

In this section, the proposed method is evaluated with numerical simulations. The considered femtocell networks in the simulations are generated based on the model used in [126], i.e. a certain number of APs randomly deployed in a fixed square region. The simulations involve six femtocell networks with different femtocell numbers. The topologies of these generated femtocell network are shown in Fig. 21. For all the six networks, there are total six channels available, and the channel demands of the APs are randomly sampled from the set \{0, 1, 2, 3, 4\}.

To begin with, the convergence capability of the proposed multiagent Q learning algorithm is checked with the femtocell network of 30 APs. The convergence capability is verified from two aspects: 1) for a specific traffic state, whether the multiagent Q learning algorithm can generate the converged channel quality estimation; 2) with multiple training cases, whether the weights of proposed graph neural network based architecture can be trained to converged value.

To verify the first point, this work randomly generates a channel demand and assign a random channel Q value for each of the 30 APs. The multiagent Q learning algorithm of Table I is applied to update the Q value for all the APs. The total learning step is set to 400, and the exploration rate \(\epsilon\) is gradually decreased during the learning process. The evolution procedure of the Q values is shown in Fig. 22. In Fig. 22, there are six sub-figures that show the corresponding Q value for the six channels. In each sub-figure, to facilitate the visualization, we plot the difference between the value at each step \(q^t\) and its final value \(q^T\) instead
Figure 21. The communication topologies of the six femtocell networks [5]. The red circles in each subfigure denote the APs and the edges represent the interference relationship.

of plotting $q'$ directly. From this figure, we can clearly observe that through the multiagent Q learning, each AP can obtain a converged quality estimation for all channels.

For the second point, 1000 traffic states of the same femtocell network are generated for the training purpose and are applied to train the proposed architecture with the multiagent Q learning algorithm of Table 9. The applied GCN is designed to having four layers with 16 convolutional channels in each layer. So we have $\Theta^1 \in \mathbb{R}^{1 \times 16}$, $\Theta^2 \in \mathbb{R}^{16 \times 16}$, $\Theta^3 \in \mathbb{R}^{16 \times 16}$, and $\Theta^4 \in \mathbb{R}^{16 \times 16}$. For each AP, the fully connected network that maps the vertex feature to the Q value is set to a three-layer neural network with 16 hidden neurons. Both $f_g(\cdot)$ and $f_q(\cdot)$ are set to
the leaky Rectifier activation function:

\[
\text{Relu}(x) = \begin{cases} 
  x, & x \geq 0 \\ 
  \beta x, & x < 0
\end{cases}
\]  

(67)

where \( \beta \) is the negative slope coefficient and set to 0.1 in our work. To verify that this architecture can be trained to the converged weights, the training error \((e)\) and the value of \(\Theta^1\) are plotted in Fig. 23. Evidently, both the training error and the weights converge during the training. It should be noted that we just illustrate the weights of the first graph convolutional layer for a concise visualization as the weights in other layers have too many components.

Then, the performance of the proposed method is evaluated on all femtocells networks. For each network, 1000 traffic states are randomly generated for training purpose and another 2000 traffic states are generated for testing. The proposed
model for each femtocell is built with the same parameter settings of the previous one used for the convergence demonstration. After being trained by the multiagent Q learning algorithm of Table 9 on all training cases, these proposed architectures are applied to solve the corresponding testing cases. Naturally, a good performance is indicated by a low interference level for each AP. Thus, a metric that measures the average interference level is defined as

\[
T = \frac{1}{L} \sum_{j=1}^{L} \sum_{n=1}^{N} \frac{1}{D_{n,j}} \sum_{C_{n,j} \in a_{n,j}} I_{n,i,j},
\]  

(68)
In (68), $L$ represents the total testing volume which is 2000 in this simulation; $j$ is the testing cases index; the quantities involve $j$ indicate that their value is based on the $j$-th testing case. The optimal value of $I$ can be obtained by solving the following mixed-integer quadratic programming (MIQP) problem for every testing case $j (1 \leq j \leq L)$, which is

$$\text{Minimize} \quad \sum_{n=1}^{N} \frac{1}{D_{n,j}} \sum_{m=1}^{M} x_{n,m}[\sum_{k \in J_n} x_{k,m}]$$

$$\text{Subject to} \quad \sum_{m=1}^{M} x_{n,m} = D_n, \forall n$$

$$x_{n,m} \in \{0, 1\}, \forall n, \forall m$$

In this formulation, $x_{n,m}$ is the binary decision variable of AP $n$ on channel $m$, where $x_{n,m} = 1$ if the channel is accessed; otherwise, $x_{n,m} = 0$. Obviously, solving this MIQP requires the global information of the whole network and the computation time increase exponentially as the network scales up. However, this simulation still illustrate the results of MIQP as the upper limits. To make a better evaluation, the proposed method is also compared with the autonomous best response algorithm proposed in [34]. This algorithm generates the solution through an iterative process. At each iteration, a subset of none neighboring APs will be randomly selected to update their channel access policy with the best response action. Moreover, this simulation also shows the performance of the random selection policy to provide a baseline for these two methods, which will randomly select channels for all APs. The simulations results concerning the average interference level of all these four methods are shown in Fig. 24 (a).

It can be seen that the average interference levels under both the autonomous best response algorithm and our proposed methods are significantly lower than that achieved by the random selection policy. Solving MIQP will lead us to the best results in terms of $T$. However, the computation time can be tremendous for the large femtocell networks, which is the most critical disadvantage of this approach. In our experiments, we solve the MIQP with CPLEX [128] and it will
take several days to get the converged results when $N = 60$. Under this situation, we limit the optimizing time of CPLEX for solving the 2000 testing cases of each femtocell network to two hours, and the corresponding results are illustrated with the green bars in Fig. 24 (a). It can be observed that the MIQP can achieve a certain advantage over our method in terms of $I$. But, the advantage appears to be not significant enough considering the huge computation time. Note that this advantage shrinks as the network scales up. The reason behind this phenomenon is that within the limited time, CPLEX is likely to find the optimal solutions for the small femtocell networks; however, for a large femtocell network, the solutions found by CPLEX may not converge to the optima yet. Furthermore, the MIQP is a centralized solution that requires the global information of the femtocell networks. In contrast, our proposed method and the autonomous best response algorithm can yield the channel access schemes based on the local information.

With the autonomous best response algorithm, a marginally lower average interference level can be obtained comparing with the proposed method, which means the target competitor performs better in terms of solution quality. However, this slight improvement is based on the cost of longer computation time. The computation time of these two methods for solving the 2000 testing cases is depicted in Fig. 7 (b). It should be noted that the computation times of the random police and MIQP are not compared. For the random policy, almost no computation is involved so that the computation time can be neglected. For the MIQP, the magnitude of computation time to get a converged solution is on a much higher order, e.g., several days when $N = 60$. From Fig. 24 (b), we can see that when the femtocell network is small ($N = 10$), the computation time of our method is a little longer than that of the autonomous best response algorithm; however, for larger femtocell networks, the proposed method cost obviously shorter computation time.
Furthermore, it can be envisioned that this advantage on computation time will get larger as the femtocell network scales up. This is because the autonomous best response algorithm needs more iterations to converge as the AP number in a femtocell network increase. For instance, the autonomous best response algorithm can converge in about five iterations when it is applied to the femtocell network with ten APs, while for the femtocell network with 60 APs, it requires 30 iterations to generate a converged solution. This results evidently demonstrate the superiority of our method on the DSA of large femtocell networks.

5.6 Chapter Summary

In this chapter, we investigate the DSA problem for femtocell networks. Traffic loads in each femtocell are assumed to be different, which is characterized by different channel demands. A graphical game is employed to coordinate the channel access among the APs. A GNN-based multiagent RL approach is designed for this problem. With the proposed method, each AP aggregates information from its neighbors through a graph neural network and based on this information, it estimates the channel quality via a local fully connected neural network. Extensive simulations on femtocell networks of different scales demonstrate that comparing with the existing autonomous best response algorithm, our method can generate the solutions of nearly the same quality while achieving advantages on the computation time, especially for the femtocell network with a quantity of APs. It can be envisioned that our method has great application potential on DSA for large scale femtocell networks.
Figure 24. Performance comparison under different femtocell networks [5]. In (a), x-axis denotes the number of APs of the femtocell networks, and the height of the bar measured by y-axis announces the average interference level of the solution generated by the corresponding method. This results for each femtocell network is based on the average of the 2000 testing cases. In (b), the pie charts show the computation time (in seconds) of the autonomous best response algorithm and the proposed graph neural network based reinforcement learning approach.
CHAPTER 6
Conclusion and Future Work

6.1 Conclusion

This dissertation first focuses on the GNN-based representation learning for graph-structured data under the semi-supervised learning framework. Most of the previous GNN-based methods depend on the positive correlation of the connected nodes. While, in this research, we introduce a feature extraction framework that inference the graph features based on the negative correlations among nodes. This framework contains a negative link generation model that discovers the negatively correlated node pairs in a graph. Moreover, to avoid searching neighbors from the whole dataset, which is of high computation cost, a high-efficiency training algorithm is designed, which randomly searches negative neighbors in a small subset of nodes. An optimization algorithm is also designed to enable the generated negative links to improve the semi-supervised learning performance. Thorough experiments have been conducted on three benchmark datasets. Results demonstrate that the proposed method is able to outperform or match the performance of state-of-the-art models in most settings.

This dissertation then investigates the energy efficiency optimization for distributed cooperative spectrum sensing of cognitive radio networks. By analyzing the mechanism of distributed cooperative spectrum sensing, an energy efficiency optimization framework is formulated. This dissertation has designed a sequential sensor selection process to find a legitimate subset of sensors that can not only fulfill the target sensing performance but also guarantee the topology requirement of the distributed sensing algorithm. A GNN-based model is introduced to make the decision for each step during the sensor selection process. RL is applied to train the proposed model to make an accurate evaluation of each action. Simu-
lation results under different network scales demonstrate the effectiveness of the proposed approach.

GNN is also applied to the interference estimation problem for D2D communication networks. In the considered network, the D2D users share common spectrum resources and there is a base station (BS) that functions as the centralized controller to coordinate the spectrum access of the whole network. The D2D network change dynamically, which means a D2D pair will terminate the D2D link when they finish the data transmission and new D2D communication requests can occur over time. To estimate the interference for the new-arrival D2D users efficiently, this dissertation has developed a GNN-based interference estimation framework. With the proposed method, the BS can estimate the interference information for a new D2D pair based on their locations. A training algorithm is also designed to optimize the parameters of the proposed model. Experiment results show the effectiveness of the developed method.

This dissertation also proposes a GNN-based DSA scheme for femtocell networks. An RL framework has been formulated for the DSA of femtocell networks. With this approach, a desired spectrum access policy can be automatically discovered. The learned policy can instruct each AP to directly find the channels which will lead to low interference. A GNN-based model has been developed to implement the proposed reinforcement learning. In this way, the learned policy is embedded in the model and possesses excellent generalization capability, which can be directly applied to each encountered traffic state. Moreover, the trained GNN-based model can make the decision for each AP in a distributed style since it generates channel access schemes based on local information. Extensive simulations on femtocell networks of different scales demonstrate that comparing with the existing algorithms, the method can generate solutions of nearly the same quality.
while achieving obvious advantages in terms of computation time, especially for the femtocell network with a quantity of APs.

6.2 Future Work

Based on the research experience of this work, the following investigations are considered as future research opportunities:

- Traffic pattern analysis for wireless communication systems. The spatial and temporal traffic patterns of a wireless communication network are significant for the planning and resource management of the system. In current literature, the traffic patterns of a communication network are assumed to be static or follow some simplified statistical model. It is interesting to apply advanced machine learning techniques to build more comprehensive models for the traffic patterns of wireless communication networks.

- Security, privacy, and stability in machine learning assisted wireless communication. In most of the recent researches that apply machine learning algorithms to communication networks, more user information is required than the traditional approaches. In this case, how to keep the privacy of the users is a critical consideration. Moreover, when deep neural networks are applied to control or optimize a system, the stability issue is important as the behavior of deep neural networks can be difficult to analyzed.

- Hardware Implementation for the advanced machine learning model. Currently, numerous advanced machine learning models have been applied to solve the problems of communication systems and most of them are evaluated through numerical simulations. As a real-world application can require more than mathematical correctness, e.g. the computational efficiency and robustness, it is necessary to know whether the developed models can meet
the expectation on a real-world system.
LIST OF REFERENCES


BIBLIOGRAPHY


