

# Wireless D2D Network Link Scheduling based on Graph Embedding

Jingyun Fu\*, Ning Ma<sup>†</sup>, Mingqiao Ye<sup>‡</sup>, Mengyuan Lee\*, Guanding Yu\*

\*College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China

E-mail: {fujingyun, mengyuan\_lee, yuguanding}@zju.edu.cn

<sup>†</sup>College of Computer Science and Technology, Zhejiang University, Hangzhou, China

E-mail: 3170101236@zju.edu.cn

<sup>‡</sup>College of Electrical Engineering, Zhejiang University, Hangzhou, China

E-mail: ymq2017@zju.edu.cn

**Abstract**—Wireless link scheduling in D2D communication systems aims at maximizing the weighted sum rate of D2D pairs by determining which subset of D2D pairs should be activated. However, it is a non-convex combinatorial optimization problem, which is generally NP-hard and difficult to achieve the optimal solution. Inspired by the recent attempt of introducing machine learning and graph embedding to reach the general goal, we propose an efficient method to solve the weighted sum rate maximization problem. We first model the system as a one-nearest neighbor graph, in which each D2D pair is a node and the strongest interference link for each node is an edge. Then we compute the feature vectors of both weights and nodes by graph embedding and use the feature vectors as the input of a subsequent multi-layer classifier. The parameters of classifier and graph embedding are trained jointly in a supervised manner. Simulation result shows that the proposed method can obtain near-optimal performance with only hundreds of training samples and is capable to be generalized to more complicated scenarios.

## I. INTRODUCTION

Efficient link scheduling in device-to-device (D2D) communication systems is an important and promising issue [1], [2]. It attempts to decide which subset of mutually interfering D2D pairs in the network to be activated to achieve the maximum utilization of the network resource. However, it is a challenging issue and up to now there is no efficient global optimal algorithm. The traditional methods [3]–[7] are all sub-optimal and need the help of channel state information (CSI) that is obtained via effective channel estimation and feedback. Thus these methods generally have high computational complexity for large-scale networks.

Recently, machine learning (ML) technique has been used to overcome the aforementioned problems. For example, the authors in [8] have used a new method, spatial learning, for wireless link scheduling, which does not require CSI. However, the training process may fail to converge if incorporating the weights as an extra input to the neural network. Moreover, hundreds of thousands of training samples is needed according to [8] and the training process is very time-consuming. To deal with these problems, IGCNet model, a model based on graph neural networks, has been proposed in [9]. However, the ML method in [9] still requires CSI and thousands of training samples. On the other hand, a graph embedding based

method has been proposed in [10], where accurate CSI is not necessary and only hundreds of training samples can lead to near-optimal results. However, the work in [10] cannot be extended into the scenario where the weight of each D2D pair is taken into consideration. Therefore, the case where D2D pairs have different weights is quite challenging and more efficient algorithms are expected to be developed.

In this paper, we go one step further to solve the wireless link scheduling problem with changeable weights. To achieve this goal, we first model the entire system as a one-nearest neighbor graph, in which each D2D pair is a node. One-nearest graph means, for each node, only the interference from the nearest transmitter is taken into consideration and interference from other transmitters is ignored. In other words, only the strongest interference links are modeled as edges. Then both node features and weights are embedded into low-dimensional vectors via two independent graph embedding processes, respectively. These feature vectors are then input into a multi-layer classifier, which will output the activation probability of each node. Simulation result shows that the proposed method can achieve near-optimal results with small-scale training samples and outperforms the directed fully-connected graph model in [10].

The remainder of this paper is organized as follows. The problem formulation and the one-nearest neighbor graph model are introduced in Section II. Section III introduces the solving process for the wireless link scheduling problem with weighted sum rate based on graph embedding and ML. The test results and performance analysis of the proposed method are presented in Section IV. Finally, we conclude this paper in Section V.

## II. PROBLEM FORMULATION AND ONE-NEAREST NEIGHBOR GRAPH MODEL

In this section, we will introduce the system model and problem formulation, and then describe the one-nearest neighbor graph model for the D2D communication system.

### A. System Model

We consider a D2D network as shown in Fig. 1.  $L$  D2D pairs in a set  $\mathcal{D} = \{D_1, \dots, D_L\}$  are randomly located in a two-dimensional plane area, where each D2D pair consists of

a transmitter  $T_i$  and a receiver  $R_i$ . The distance between  $T_i$  and  $R_i$  is denoted as  $d_i$ , which varies in the range of  $d_{\min}$  to  $d_{\max}$ . When  $D_i$  is activated, the transmission power of  $D_i$  is fixed as  $p_i$  and we set its corresponding activation indicator  $\rho_i = 1$ . Otherwise, if the  $i$ -th D2D pair is not activated,  $\rho_i = 0$ . Besides,  $h_{ii}$  denotes the channel power gain of  $D_i$  and  $h_{lk}$  denotes the interference channel power gain from  $D_l$  to  $D_k$ .

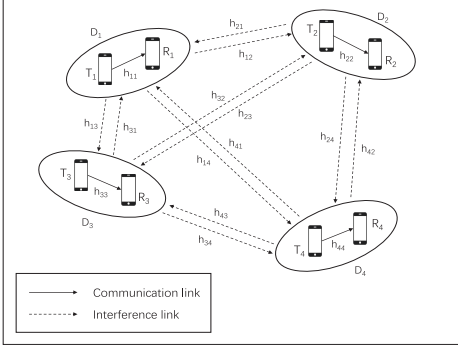


Fig. 1. D2D communication system.

Accordingly, the signal-to-interference-plus-noise ratio (SINR) of  $D_i$  is defined as

$$SINR_i(\rho) = \frac{\rho_i p_i |h_{ii}|^2}{\sigma_N^2 + \sum_{k \neq i} \rho_k p_k |h_{ki}|^2}, \quad (1)$$

where  $\sigma_N^2$  denotes the power of the additive white Gaussian noise (AWGN). Moreover,  $\rho = [\rho_1, \rho_2, \dots, \rho_L]$ . Correspondingly, the data transmission rate of  $D_i$  on the bandwidth  $B$  can be expressed as follows

$$R_i(\rho) = B \log(1 + SINR(\rho)), \quad (2)$$

$$= B \log\left(1 + \frac{\rho_i p_i |h_{ii}|^2}{\sigma_N^2 + \sum_{k \neq i} \rho_k p_k |h_{ki}|^2}\right). \quad (3)$$

According to (3), if too many D2D pairs are activated at the same time, the interference between the links will be very serious, thus reducing the overall data transmission rate. In addition, different D2D pairs are allowed to have different weights in this model. Taking weight into consideration has important practical significance, because D2D pairs may have different priorities in practical communication system. In this way, the link scheduling problem can be formulated as

$$\max_{\rho} \sum_{i=1}^L \omega_i R_i(\rho), \quad (4)$$

s.t.

$$\rho_i \in \{0, 1\}, \forall i, \quad (4a)$$

where  $w_i$  is the weight of the  $i$ -th D2D pair.

### B. One-nearest Neighbor Graph Model

We use  $G(V, E, \mathbf{x}, \mathbf{w}, \alpha(u, v))$ , a one-nearest neighbor graph in Fig. 2, to represent the system in Fig. 1. In this graph,  $V$  is the set of nodes and  $E$  is the set of edges. Specifically, we regard each D2D pair as a node and the

edges connect different D2D pairs. In the one-nearest neighbor graph, only interference from the nearest transmitter is taken into consideration for each receiver. In this way, the edge set,  $E$ , only contains  $L$  elements and each edge in  $E$  is directed. Compared with the fully-connected graph in [10], one-nearest graph can effectively reduce the computational complexity of the following graph embedding process. Besides,  $\mathbf{x}$  is the feature of each node. Specifically, we use inner distance between  $T_i$  and  $R_i$  as the feature for node  $D_i$ . Moreover,  $\mathbf{w}$  is the set of D2D pairs' weights and  $\alpha(u, v)$  is the distance between node  $u$  and  $v$ , for  $u, v \in V$ .

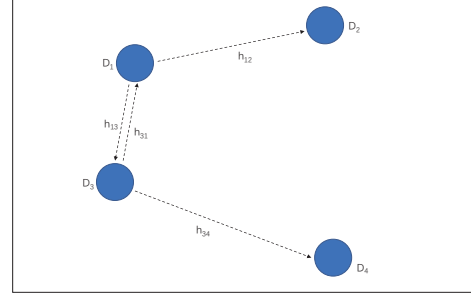


Fig. 2. One-nearest neighbor graph model.

## III. GRAPH EMBEDDING BASED METHOD FOR WIRELESS LINK SCHEDULING WITH WEIGHTED SUM RATE

In this section, we will introduce how to deal with the link scheduling problem (4) with graph embedding method.

### A. Graph Embedding Process

Graph embedding [11] can effectively use low-dimensional vectors to represent the complex graph information. In this paper, we use *structure2vec* [12], a deep learning architecture for graph embedding, to calculate the  $p$ -dimensional feature vectors for each node and its weight. We denote  $\mu_v^{(t)}$  and  $\mu_{vw}^{(t)}$  as the embedding vectors for each node  $v \in V$  and each node's weight, respectively. According to *structure2vec*, they are updated using the following two independent iterations,

$$\mu_v^{(t+1)} = \Gamma\left(\mathbf{x}_v, \{\alpha(u, v)\}_{u \in N(v)}, \{\mu_u^{(t)}\}_{u \in N(v)}\right), \quad (5)$$

and

$$\mu_{vw}^{(t+1)} = \Gamma\left(\mathbf{x}_{vw}, \{\alpha(u, v)\}_{u \in N(v)}, \{\mu_{uw}^{(t)}\}_{u \in N(v)}\right), \quad (6)$$

respectively, where  $\Gamma$  is a nonlinear function,  $N(v)$  is the set of nodes adjacent to  $v$ ,  $\mathbf{x}_v$  is the node feature and  $\mathbf{x}_{vw}$  is the weight feature. As mentioned above, our graph is a one-nearest graph. Therefore,  $N(v)$  only contains the nearest node to  $v$ . As shown in [12], the information of one node can propagate farther with more iterations. We follow the setting of the number of iterations  $T = 2$  in [10] since the general graph architecture is similar. By iteratively updating  $\mu_v^{(t)}$  and  $\mu_{vw}^{(t)}$  using (5) and (6), the feature vectors for each node and its weight can be obtained based on the topology of the one-nearest neighbor graph.

Specifically,  $\mathbf{x}_v$  is the D2D pair's inner distance and  $\mathbf{x}_{vw}$  is the D2D pair's weight. In addition, we use the rectified linear

unit (ReLU) as nonlinear function mapping and implement the *structure2vec* architecture in embedded mean field. So the iteration rules are given by

$$\mu_v^{(t+1)} = \sigma \left( \mathbf{W}_1 \mathbf{x}_v + \mathbf{W}_2 \sum_{u \in N(v)} \alpha(u, v) + \mathbf{W}_3 \sum_{u \in N(v)} \mu_u^{(t)} \right), \quad (7)$$

and

$$\mu_{vw}^{(t+1)} = \sigma \left( \mathbf{W}_4 \mathbf{x}_{vw} + \mathbf{W}_5 \sum_{u \in N(v)} \alpha(u, v) + \mathbf{W}_6 \sum_{u \in N(v)} \mu_{uw}^{(t)} \right), \quad (8)$$

where  $\sigma(x) = \max(0, x)$ , and the set  $\mathcal{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_6\}$  contains the parameters for different input features. The pseudo codes for the two independent graph embedding processes are listed in Table I and Table II, respectively.

According to [10], quantification of weights and distances is required to fit the *structure2vec* network architecture. Thus we use  $q$ -bit one-hot feature to represent the quantified result. Specifically, the quantizer range for weight is  $[w_{\min}, w_{\max}]$ , the quantizer range for inner distance of D2D pairs is  $[d_{\min}, d_{\max}]$  and the quantizer range for interference links is  $[0, d_{\text{area}}]$ . The impact of the numbers of quantization bits will be discussed in Section IV.

TABLE I  
GRAPH EMBEDDING PROCESS FOR NODE FEATURES

**Algorithm 1** Embedded Mean Field for Node

```

1: input:  $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$ 
2: initialization  $\mu_v^{(0)} = 0$ , for all  $v \in V$ .
3: for  $t = 1$  to  $T$  do
4:   for  $v \in V$  do
5:      $\mu_v^{(t+1)} = \sigma \left( \mathbf{W}_1 \mathbf{x}_v + \mathbf{W}_2 \sum_{u \in N(v)} \alpha(u, v) + \mathbf{W}_3 \sum_{u \in N(v)} \mu_u^{(t)} \right)$ 
6:   end for
7: end for
8: return  $\{\mu_v^{(T)}\}_{v \in V}$ .
```

TABLE II  
GRAPH EMBEDDING PROCESS FOR WEIGHT

**Algorithm 2** Embedded Mean Field for Weight

```

1: input:  $\mathcal{W} = \{\mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6\}$ 
2: initialization  $\mu_{vw}^{(0)} = 0$ , for all  $v \in V$ .
3: for  $t = 1$  to  $T$  do
4:   for  $v \in V$  do
5:      $\mu_{vw}^{(t+1)} = \sigma \left( \mathbf{W}_4 \mathbf{x}_{vw} + \mathbf{W}_5 \sum_{u \in N(v)} \alpha(u, v) + \mathbf{W}_6 \sum_{u \in N(v)} \mu_{uw}^{(t)} \right)$ 
6:   end for
7: end for
8: return  $\{\mu_{vw}^{(T)}\}_{v \in V}$ .
```

### B. Multi-layer Classifier

After the feature vectors of the one-nearest neighbor graph are generated via graph embedding, the next step is to decide which subset of  $V$  should be activated to achieve the optimal network resource utilization. This is a binary classification problem and we use a multi-layer classifier shown in Fig. 3 to solve it.

As depicted in Fig. 3, both feature vectors of the nodes and the weights are taken as input to the classifier network. Therefore, the input layer consists of  $2p$  neurons corresponding to the two  $p$ -dimensional feature vectors  $\mu_{vw}^{(t)}$  and  $\mu_v^{(t)}$ . The output layer consists of two neurons that represent the activation probability for each node.

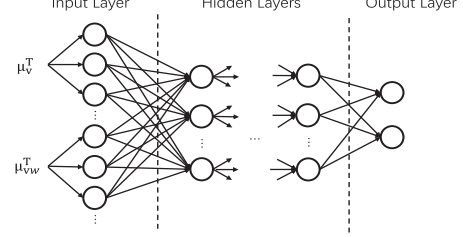


Fig. 3. Multi-layer classifier.

### C. Training Process

Since the integral network consists of graph embedding and multi-layer classifier, we use the discriminative training method to jointly learn the parameters of graph embedding  $\mathcal{W} = \{\mathbf{W}_1, \dots, \mathbf{W}_6\}$  and classifier  $F$  in a supervised manner. The training dataset is generated according to the FPLinQ algorithm in [6]. The training dataset is defined as  $\Gamma = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$ , where  $\mathbf{x}_n$  refers to the input of the original graph and  $\mathbf{y}_n$  is the link scheduling result, which is denoted as an  $L$ -dimensional boolean vector.  $\mathbf{y}_n(l) = 1$  means  $D_l$  is activated and  $\mathbf{y}_n(l) = 0$  means  $D_l$  is not activated. Moreover, the original output of classifier is denoted as  $\mathbf{z}_l^n$ . To be specific,  $\mathbf{z}_l^n = (0, 1)$  when  $\mathbf{y}_n(l) = 1$  and  $\mathbf{z}_l^n = (1, 0)$  when  $\mathbf{y}_n(l) = 0$ . In addition,  $F(\mu_v^{(n)})$  is denoted as  $\tilde{\mathbf{z}}_l^n$ . We use cross entropy as the loss function and thus the graph embedding and classifier  $F$  can be learned together via the following optimization problem

$$\min_{\mathbf{W}, F} \sum_{n=1}^N \sum_{l=1}^L -\mathbf{z}_l^n(0) \log \tilde{\mathbf{z}}_l^n(0) - \mathbf{z}_l^n(1) \log \tilde{\mathbf{z}}_l^n(1). \quad (9)$$

## IV. TEST RESULTS

In this section, we will test the performance of the proposed method based on the open source *structure2vec* code<sup>1</sup>. FPLinQ algorithm is implemented by MATLAB and the rest is implemented by python 3.7.

### A. Network Setup

As shown in Fig. 1, a 500 m by 500 m two-dimensional region with 50 D2D pairs is considered. Within this area, the transmitters are randomly distributed and the corresponding receivers are also randomly distributed within a range of 2-65 m from the transmitter. The short-range outdoor model ITU-1411 [13] is used as channel model which only considers the distance-dependent path-loss. Detailed parameters for D2D networks are listed in Table III unless otherwise stated.

The number of iterations  $T$  is set to 2 according to Section III. We adopt a four-layer classifier whose input is the graph

<sup>1</sup>[https://github.com/Hanjun-Dai/pytorch\\_structure2vec/tree/master/s2v\\_lib](https://github.com/Hanjun-Dai/pytorch_structure2vec/tree/master/s2v_lib)

TABLE III  
SIMULATION PARAMETERS

Parameter	Value
Edge length, $d_{\text{area}}$	500 m
D2D distance, $d_{\text{min}}, d_{\text{max}}$	2 m, 65 m
Noise spectral density	-169 dBm/Hz
Bandwidth, $B$	5 MHz
Carrier frequency	2.4 GHz
Antenna height	1.5 m
Transmit power of activation link, $P_l$	40 dBm/Hz

embedding result. Meanwhile, the dimensions of the feature vectors for the pairwise distance and the weight of the D2D pairs are both set as 16. Therefore, the total size of input layer of the classifier is 32. Then, we set the size of the two hidden layers to 32 and 16, respectively. Besides, the output layer's dimension is set to 2. We adopt the rectified linear unit as the activation function for the classifier and we also adopt batch normalization to avoid the vanishing gradient problem and the overfitting problem. Moreover, we use the adaptive moment estimation (Adam) [14] algorithm as the optimization algorithm. Detailed parameters for graph embedding and classifier are listed in the Table IV unless otherwise stated.

TABLE IV  
NETWORK PARAMETERS

Parameter	Value
Number of iterations, $T$	2
Pairwise distance embedding dimension, $P_1$	16
Weight embedding dimension, $P_2$	16
First hidden layer size, $H_1$	32
Second hidden layer size, $H_2$	16
Number of training samples	500
Number of testing samples	50
Distance quantization bit, $q_1$	3
Weight quantization bit, $q_2$	3
Batch size	128

### B. Impact of the Number of Training Samples

Since large-scale training samples are generally hard to obtain, the required number of training samples is critical in the training stage of neural networks. This is the reason why we attempt to design a network which can perform well with only a relatively small number of training samples.

We test the performance of the proposed method with different numbers of training samples. Two indicators, classifier accuracy and average sum rate, are tested. The classifier accuracy refers to the accuracy of supervised learning. And the average sum rate is the ratio of the objective function value given by our proposed method with respect to that of the FPLinQ algorithm.

As shown in Table V, there are some fluctuations for the two indicators as the number of training samples increases. For the classifier accuracy, it increases slightly with the number

TABLE V  
PERFORMANCE WITH DIFFERENT NUMBERS OF TRAINING SAMPLES

Number of training samples	200	500	1000	1500	2000
Classifier accuracy	0.8252	0.8384	0.8396	0.8476	0.8428
Average sum rate	0.9698	0.9797	0.9733	0.9817	0.9776

TABLE VI  
PERFORMANCE ON SCENARIOS WITH DIFFERENT NUMBERS OF D2D PAIRS

Number of D2D pairs	10	30	50	80	100
Classifier accuracy	0.9220	0.8407	0.8384	0.8465	0.8530
Average sum rate	0.9715	0.9810	0.9797	0.9639	0.9603

TABLE VII  
PERFORMANCE ON SCENARIOS WITH DIFFERENT PAIRWISE DISTANCES

Pairwise distance $d_{\text{min}} - d_{\text{max}}$ (m)	2-65	10-50	30-70	all 30
Classifier accuracy	0.8384	0.8032	0.7592	0.7508
Average sum rate	0.9797	0.9617	0.9892	0.9438

of training samples. For the average sum rate, our proposed method can achieve 97.97% of the performance by the FPLinQ [6] when the size of the training set is 500. Besides, the average sum rate keeps almost unchanged as the sample size further increases. When the sample size increases to 1500, the average sum rate only increases by 0.20%. These results suggest the proposed method only needs a few hundred training samples to achieve a relatively high average sum rate.

### C. Scalability Test

Intuitively, the performance of the ML algorithm will deteriorate in complex scenarios. Therefore, the performance of our proposed method with different numbers of D2D pairs is tested. The test results are listed in Table VI.

From Table VI, the classifier accuracy is relatively high when there are only 10 D2D pairs in the system. However, the classification accuracy drops to about 84% when the number of D2D pairs is greater than 30. This is because the features are easy to learn when the topology of the graph is simple but the performance would get worse when the topology of the graph becomes more complicated. Meanwhile, the average sum rate of our proposed algorithm only decreases by 1.58% when the number of D2D pairs changes from 50 to 80. And the average sum rate only decreases by 0.36% when the number of D2D pairs increases from 80 to 100. This indicates that the proposed method has a good generalization ability for the complexity of the graph topology.

Next we test the performance of the proposed method with different pairwise distances of D2D pairs and the test results are presented in Table VII. Data shows that the performance of the proposed method deteriorates as the pairwise distance distribution interval decreases. Here we can give some intuitive explanations. The distinction between nodes will be reduced as the range of pairwise distance distribution gets smaller. Thus the smaller the allowed range of the distance distribution is, the more difficult for the proposed method to extract features from the graph. Therefore, both the classifier accuracy and average sum rate of our proposed algorithm decrease when the range of distance distribution gets smaller.

### D. Influence of the Numbers of Quantization Bit

As mentioned in Section III, distances and weights of each node have been quantized to fit the *structure2vec* architecture. On the basis of the analysis above, it is noted that the diversity of node features is an important factor which would influence

TABLE VIII  
PERFORMANCE WITH DIFFERENT NUMBERS OF DISTANCE  
QUANTIZATION BIT

Number of Quantization Bit	2	3	4	5
Classifier accuracy	0.8064	0.8384	0.8700	0.8860
Average sum rate	0.9535	0.9797	0.9899	0.9697

TABLE IX  
PERFORMANCE WITH DIFFERENT NUMBERS OF WEIGHT QUANTIZATION  
BIT

Number of Quantization Bit	2	3	4	5
Classifier accuracy	0.8356	0.8384	0.8368	0.8440
Average sum rate	0.9750	0.9797	0.9729	0.9696

the performance of the proposed algorithm. Consequently, the quantization bits of distance and weight,  $q_1$  and  $q_2$ , may also have an important impact on the performance of the proposed algorithm. Thus we test the influence of  $q_1$  and  $q_2$ , respectively.

Firstly, we only change the distance quantization bit,  $q_1$ . The results are shown in Table VIII. The classifier accuracy increases with the increase of  $q_1$ . This is because the diversity of distance gets more apparent when there are more distance quantization bits. Moreover, since quantization is always accompanied by errors, using more quantization bits means less information loss and the quantized results are closer to the original values in general. However, the average sum rate increases first and then decreases according to Table VIII. This indicates that there is a threshold for distance quantization bit as shown in [10]. Noted that achieving a higher average sum rate is our ultimate goal, we need to pay special attention to the existence of this threshold when selecting the suitable  $q_1$ .

Secondly, we only change the weight quantization bit,  $q_2$ , and the results are shown in Table IX. The test result indicates that the influences of  $q_2$  and  $q_1$  on the performance of the proposed algorithm are basically the same. The classifier accuracy shows an upward trend, while the average sum rate has a threshold. However, the influence of the weight quantization bit on the performance of the proposed algorithm is slightly worse than that of the distance quantization bit, which can be explained from several aspects. On the one hand, distance quantization is used more frequently than weight quantization in the proposed algorithm. To be more specific, weight quantization exists only within the D2D pairs, while distance quantization exists both within and between the D2D pairs. On the other hand, the weight itself is actually a relative value but distance is a specific value. Furthermore, since distance itself implies an activation priority, distance contains more information than weight.

## V. CONCLUSION

This paper aims at solving the wireless link scheduling problem where the weight of each D2D pair is taken into consideration. The key idea is to construct a one-nearest neighbor graph model for the D2D communication system to reduce the computational complexity and generate feature vectors for both node and weight by two independent graph embedding processes. Thus the graph embedding result reflects the topology and interference patterns of the network. Then we use a multi-layer classifier to output the activation probability

of each D2D pair and decide which subset of D2D pairs should be activated. Simulation result shows that the proposed methods can obtain near-optimal solutions for weighted sum rate maximization problem with only hundreds of training samples.

This paper envisions the broad application prospect of machine learning in the field of wireless communications. Moreover, the test result suggests that graph embedding is an effective method for extracting features based on network topology and it may have potential for other wireless resource allocation problems.

## REFERENCES

- [1] Y. Wu, J. Chen, L. Qian, J. Huang, and X. Shen, "Energy-aware cooperative traffic offloading via device-to-device cooperations: An analytical approach," *IEEE Trans. Mobile Comput.*, vol. 16, no. 1, pp. 97-114, Jan. 2017.
- [2] Y. Wu, J. Wang, L. Qian, and R. Schober, "Optimal power control for energy efficient D2D communication and its distributed implementation," *IEEE Commun. Lett.*, vol. 19, no. 5, pp. 815-818, May. 2015.
- [3] X. Wu, S. Tavildar, S. Shakkottai, T. Richardson, J. Li, R. Laroia, and A. Jovicic, "FlashLinQ: A synchronous distributed scheduler for peer-to-peer ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1215-1228, Aug. 2013.
- [4] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: A new approach for spectrum sharing in device-to-device communication systems," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1139-1151, Jun. 2014.
- [5] X. Yi and G. Caire, "Optimality of treating interference as noise: A combinatorial perspective," *IEEE Trans. Inf. Theory*, vol. 62, no. 8, pp. 4654-4673, Aug. 2016.
- [6] K. Shen and W. Yu, "FPLinQ: A cooperative spectrum sharing strategy for device-to-device communications," *IEEE Int. Symp. Inf. Theory (ISIT)*, pp. 2323-2327, Jun. 2017.
- [7] L. Zhang, M. Xiao, G. Wu and S. Li, "Efficient scheduling and power allocation for D2D-assisted wireless caching networks," *IEEE Trans. Commun.*, vol. 64, no. 6, pp. 2438-2452, Jun. 2016.
- [8] W. Cui, K. Shen and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1248-1261, Jun. 2019.
- [9] Y. Shen, Y. Shi, J. Zhang, K. B. Letaief, "A graph neural network approach for scalable wireless power control," *arXiv:1907.08487v1 [cs.IT]*, Jul. 2019.
- [10] M. Lee, G. Yu, and G. Y. Li, "Graph embedding based wireless link scheduling with few training samples," *arXiv preprint arXiv:1906.02871*, Jul. 2019.
- [11] H. Y. Cai, V. W. Zheng, and K. C. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616-1637, Feb. 2018.
- [12] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," *Proc. Int. Conf. Mach. Learn.*, pp. 2702-2711, 2016.
- [13] *Recommendation ITU-R P.1411-8*. International Telecommunication Union, 2015.
- [14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, pp. 1-6, May 2014.