Spatial Deep Learning for Wireless Scheduling

Wei Cui, Kaiming Shen, and Wei Yu
Electrical and Computer Engineering Department
University of Toronto, Toronto, Ontario M5S 3G4, Canada
Email: {cuiwei2, kshen, weiyu}@ece.utoronto.ca

***Abstract*—The optimal scheduling of multiple interfering links in a densely deployed wireless network with full frequency reuse is a well-known challenging problem. The classical optimization approaches to this problem typically operate under the paradigm of first estimating all the interfering channel strengths then finding an optimum solution using the model. However, traditional scheduling methods are computationally and resource intensive, because channel estimation is expensive especially in dense networks, and further the optimization of link scheduling is typically a nonconvex problem. This paper takes a novel *deep spatial learning* approach to the scheduling problem. We show that it is possible to *bypass the channel estimation stage* altogether and to use a deep neural network to produce a near optimal schedule based solely on geographic locations of the transmitters and receivers in the network. This is accomplished by taking advantage of the recent advances in fractional programming that allows us to generate high-quality local optimum solutions to the scheduling problem for randomly deployed device-to-device networks as training data, and by using a novel neural network architecture that takes the *geographic spatial convolutions* of the interfering and interfered neighboring nodes as input over multiple feedback stages to learn the optimum solution.**

I. Introduction

Scheduling of interfering links is one of the most fundamental tasks in wireless networking. Consider for example a densely deployed device-to-device (D2D) network with full frequency reuse, in which nearby links produce significant interference for each other whenever they are simultaneously activated. The task of scheduling amounts to judiciously activating a *subset* of mutually "compatible" links so as to avoid excessive interference for maximizing a network utility.

The traditional approach to link scheduling is based on the paradigm of first estimating the interfering channels (or at least the interference graph topology), then optimizing the schedule based on the estimated channels. This model-based approach, however, suffers from two key shortcomings. First, the need to estimate not only the direct channels but also all the interfering channels is resource intensive. In a network of N transmitter-receiver pairs, N^2 channels need to be estimated within each coherence block. Training takes valuable resources away from the actual data transmissions; further, pilot contamination is inevitable in large networks. Second, the achievable data rates in an interfering environment are nonconvex functions of the transmit powers. Moreover, scheduling variables are binary. Hence, even with full channel knowledge, the optimization of scheduling is a nonconvex integer programming problem for which finding an optimal solution is computationally complex and is challenging for real-time implementation.

This paper proposes a new approach, named *spatial learning*, to address the above two issues. Our key idea is to recognize that the optimal link scheduling does not necessarily require the exact channel estimates, and further the interference pattern in a network is to a large extent determined by the relative locations of the transmitters and receivers. Hence, it ought to be possible to *learn* the optimal scheduling based solely on the geographical locations of the neighboring transmitters/receivers, thus bypassing channel estimation altogether. Toward this end, this paper proposes a neural network architecture that takes the geographic spatial convolution of the interfering and interfered neighboring transmitters/receivers as input, and learns the optimal scheduling in a densely deployed D2D network over multiple stages based on the spatial parameters alone.

We are inspired by the recent explosion of successful applications of machine learning techniques [1], [2] that demonstrate the ability of deep neural networks to learn rich patterns and to approximate arbitrary function mappings [3]. We further take advantage of the recent progress on fractional programming methods for link scheduling [4]–[6] that allows us to generate a large number of locally optimal solutions for random networks offline as training data. The main contribution of this paper is a specifically designed neural network architecture that facilitates the spatial learning of geographical locations of interfering and interfered nodes and is capable of achieving large portion of the optimum sum rate of the state-of-the-art algorithm in a computationally efficient manner, while requiring no explicit channel state information.

Traditional approach to scheduling over wireless interfering links for sum rate maximization are all based on (nonconvex) optimization, e.g., greedy heuristic search [7], iterative methods for achieving quality local optimum [4], [8], methods based on information theory considerations [9], [10] or hypergraph coloring [11], [12], or methods for achieving the global optimum but with exponential complexity such as polyblock-based optimization [13] or nonlinear column generation [14]. The recent re-emergence of machine learning has motivated the use of neural networks for network optimization. This paper is most closely related to the recent work of [15] in adapting deep learning to perform power control and [16] in utilizing ensemble learning to solve a closely related problem, but we go one step further than [15], [16] in that we forgo the traditional requirement of channel state information for spectrum optimization. By demonstrating that the location information (which can be easily obtained via global positioning system) can be effectively used as a proxy for obtaining near-optimum solution, we open the door for much wider application of learning theory to resource allocation problems in wireless networking.

II. Wireless Link Scheduling

Consider a scenario of N independent D2D links located in a two-dimensional region. The transmitter-receiver distance can vary from links to links. We use p_i to denote the fixed transmit power level of the ith link, if it is activated. Moreover, we use $h_{ij} \in \mathbb{C}$ to denote the channel from the transmitter of the jth link to the receiver of the ith link, and use σ^2 to denote the background noise power level. Scheduling occurs in a time slotted fashion. In each time slot, let $x_i \in \{0, 1\}$ be an indicator variable for each link i, which equals to 1 if the link is scheduled and 0 otherwise. We assume full frequency reuse with bandwidth W. Given a set of scheduling decisions x_i, the achievable rate R_i for link i in the time slot can be computed as

$$R_i = W \log \left(1 + \frac{|h_{ii}|^2 p_i x_i}{\Gamma(\sum_{j \neq i} |h_{ij}|^2 p_j x_j + \sigma^2)} \right), \tag{1}$$

where Γ is the SNR gap to the information theoretical channel capacity, due to the use of practical coding and modulation for the linear Gaussian channel [17]. The wireless link scheduling problem is that of selecting a subset of links to activate in any given transmission period so as to maximize some objective function of the achieved rates. This paper considers the objective function of maximizing the sum rate over the N users over each scheduling slot, formulated as

$$\underset{\mathbf{x}}{\text{maximize}} \quad \sum_{i=1}^{N} R_i \tag{2a}$$

$$\text{subject to} \quad x_i \in \{0, 1\}, \ \forall i. \tag{2b}$$

The overall problem is a challenging discrete optimization problem, due to the complicated interactions between different links through the interference terms in the signal-to-interference-and-noise (SINR) expressions.

This paper uses a recently developed fractional programming approach (referred to as FPLinQ) [4] to generate high-quality local optimum solutions as benchmark for the above scheduling problem. FPLinQ relies on a transformation of the SINR expression that decouples the signal and the interference terms and a subsequent coordinated ascent approach to find the optimal transmit power for all the links. The FPLinQ algorithm is closely related to the weighted minimum mean-square-error (WMMSE) algorithm for weighted sum-rate maximization [8]. For the scheduling task, FPLinQ quantizes the optimized power in a specific manner to obtain the optimized binary scheduling variables.

III. Deep Learning Based Link Scheduling

A. Learning Based on Geographic Location Information

A central goal of this paper is to demonstrate that geographical location information is already sufficient as a proxy for optimizing link scheduling. This is in contrast to traditional optimization approaches for solving (2) that require the full instantaneous channel state information, and also in contrast to the recent work [15] that proposes to use deep learning to solve the power control problem by learning the WMMSE optimization process. In [15], a fully connected neural network is designed that takes in the channel coefficient matrix as the input, and produces optimized continuous power variables as the output to maximize the sum rate. While satisfactory scheduling performance has been obtained in [15], the architecture of [15] is not scalable. In a network with N transmitter-receiver pairs, there are N^2 channel coefficients. A fully connected neural network with N^2 nodes in the input layer and N output layer would require at least $O(N^3)$ interconnect weights (and most likely much more). Thus, the neural network architecture proposed in [15] has training and testing complexity that grows rapidly with the number of links.

Instead of requiring the full set of channel state information (CSI) between every transmitter and every receiver as the input to the neural network $\{h_{ij}\}$, which has $O(N^2)$ entries, this paper proposes to use the geographic location information (GLI) as input, defined as a set of vectors $\{(\mathbf{d}_i^{\text{tx}}, \mathbf{d}_i^{\text{rx}})\}_i$, where $\mathbf{d}_i^{\text{tx}} \in \mathbb{R}^2$ and $\mathbf{d}_i^{\text{rx}} \in \mathbb{R}^2$ are the transmitter and the receiver locations of the ith link, respectively. Note that the input now scales linearly with the number of links, i.e., $O(N)$.

We advocate using GLI as a substitute for CSI, because GLI already captures the main feature of channels: the path-loss and shadowing of a wireless link are mostly functions of distance and location. In fact, accounting for fast fading in addition, the CSI can be thought of as a stochastic function of GLI

$$\text{CSI} = f(\text{GLI}). \tag{3}$$

While optimization approaches to the wireless link scheduling problem aim to find a mapping $g(\cdot)$ from CSI to the scheduling decisions, i.e.,

$$\mathbf{x} = g(\text{CSI}), \tag{4}$$

the deep learning architecture of this paper aims to capture directly the mapping from GLI to $\mathbf{x}$, i.e., to learn the function

$$\mathbf{x} = g(f(\text{GLI})). \tag{5}$$

B. Transmitter and Receiver Density Grid as Input

To construct the input to the neural network based on GLI, we quantize the continuous $(\mathbf{d}_i^{\text{tx}}, \mathbf{d}_i^{\text{rx}})$ in a grid form. Without loss of generality, we assume a square $\ell \times \ell$ meters deployment area, partitioned into equal-size square cells with an edge length of ℓ/M, so that there are M^2 cells in total. We use $(s,t) \in [1:M] \times [1:M]$ to index the cells. For a particular link i, let $(s_i^{\text{tx}}, t_i^{\text{tx}})$ be the index of the cell where the transmitter $\mathbf{d}_i^{\text{tx}}$ is located, and $(s_i^{\text{rx}}, t_i^{\text{rx}})$ be the index of the cell where the receiver $\mathbf{d}_i^{\text{rx}}$ is located. We use the tuple $(s_i^{\text{tx}}, t_i^{\text{tx}}, s_i^{\text{rx}}, t_i^{\text{rx}})$ to represent the location information of the link. We propose to construct two *density grid* matrices of size $M \times M$, denoted by T and R, to represent the density of the *active* transmitters and receivers, respectively, in the geographical area. The density grid matrices

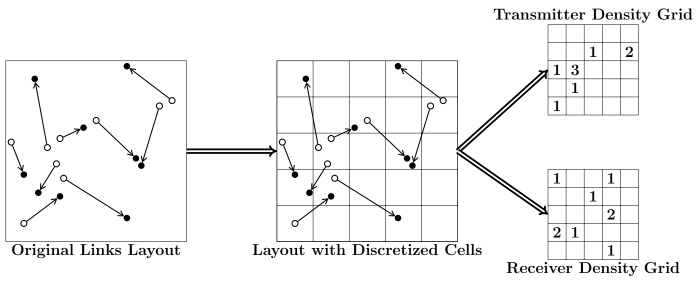


Fig. 1: Transmitter and receiver density grids.

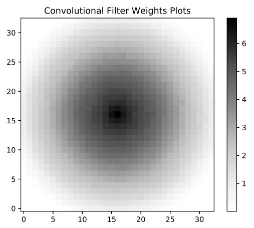


Fig. 2: Trained spatial convolution filter.

are constructed by simply counting the total number of active transmitters and receivers in each cell, as illustrated in Fig. 1. The activation pattern $\{x_i\}$ is initialized as a vector of all 1's at the beginning. As the algorithm progressively updates the activation pattern, the density grid matrices are updated as

$$T(s,t) = \sum_{\{i|(s_i^{\text{tx}},t_i^{\text{tx}})=(s,t)\}} x_i, \tag{6}$$

$$R(s,t) = \sum_{\{i|(s_i^{\text{rx}},t_i^{\text{rx}})=(s,t)\}} x_i. \tag{7}$$

C. Deep Neural Network Structure

The overall neural network structure is an iterative computation graph. A key novel feature of the network structure is a forward path including two stages: a convolution stage that captures the interference patterns of neighboring links based on the geographic location information and a fully connected stage that captures the nonlinear functional mapping of the optimized schedule. Further, we propose a novel *feedback connection* between the iterations to update the state of optimization. The individual stages and the overall network structure are described in detail below.

1) Convolution Stage: The convolution stage is responsible for computing two functions, corresponding to that of the interference each link causes to its neighbors and the interference each link receives from its neighbors, respectively. As a main innovation in the neural network architecture, we propose to use spatial convolutional filters, whose coefficients are optimized in the training process, that operate directly on the transmitter and receiver density grids described in the previous section. The transmitter and receiver spatial convolutions are computed in parallel on the two grids. At the end, two pieces of information are computed for the transmitter-receiver pair of each link: a convolution of spatial geographic locations of all the nearby receivers that the transmitter can cause interference to, and a convolution of spatial geographic locations of all the nearby transmitters that the receiver can experience interference from. The computed convolutions are referred to as TxINT$_i$ and RxINT$_i$, respectively, for link i.

Since the idea is to estimate the effect of total interference each link causes to nearby receivers and effect of the total interference each link is exposed to, we need to exclude the link's own transmitter and receiver in computing the convolutions. This is accomplished by subtracting the contributions each

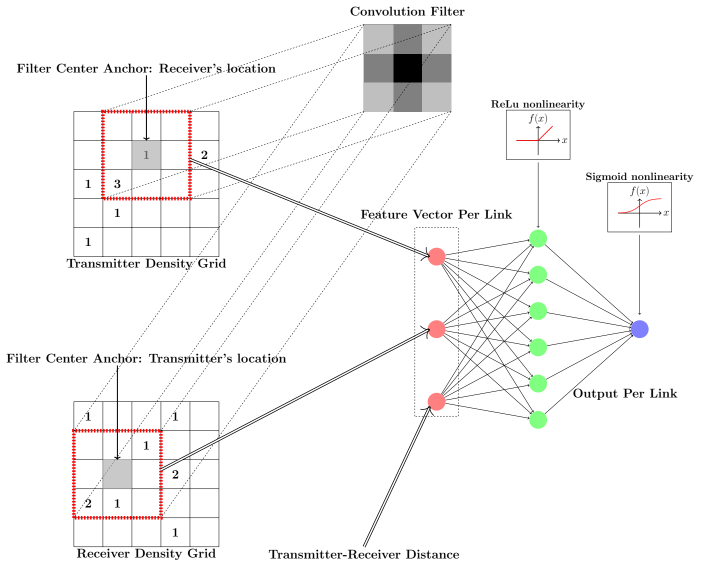


Fig. 3: Example of forward computation path for a single link with spatial convolutions and link distance as input to a neural network.

link's own transmitter and receiver in the respective convolution sum.

The convolution filter is a 2D square matrix with fixed predefined size and trainable parameters. The value of each entry of the filter can be interpreted as the channel coefficient of a transceiver located at a specific distance from the center of the filter. Through training, the filter learns the channel coefficient by adjusting its weights. Fig. 2 shows a trained filter. As expected, the trained filter exhibits a circular symmetric pattern with radial decay.

The convolution stage described above summarizes two quantities for each link: the total interference produced by the transmitter and the total interference the receiver is being exposed to. However, the interference pattern often has multiple components, coming from different neighboring links of different ranges. Thus, describing the interference pattern with more than one convolution can provide more information. Motivated by this observation, we can improve the feature vector by augmenting it with multiple convolutions. For example, at each transmitter and receiver, we can construct three convolutions filters: with small range, medium range, and full range. The three separate sets of weights of the convolution filters are tied together across all the transmitters and the receivers. In this case, TxINT$_i$ and RxINT$_i$ for each link would consist of three location components. Experimental results show that having multiple convolutions indeed improve the scheduling performance of the neural network.

2) Fully Connected Stage: The fully connected stage is the second stage of the forward computation path, following the convolution stage described above. With the feature vector extracted for each link as input, the fully connected stage produces an output in $x_i \in [0, 1]$, (which could be interpreted as optimized continuous power) for that link.

The feature vector for each link comprises of the following entries: TxINT_i, RxINT_i, and the distance between the transmitter and the receiver for this link. The tuple (TxINT_i, RxINT_i) describes the interference relation between the ith link and its neighbors, while the distance describes the link's own channel strength.

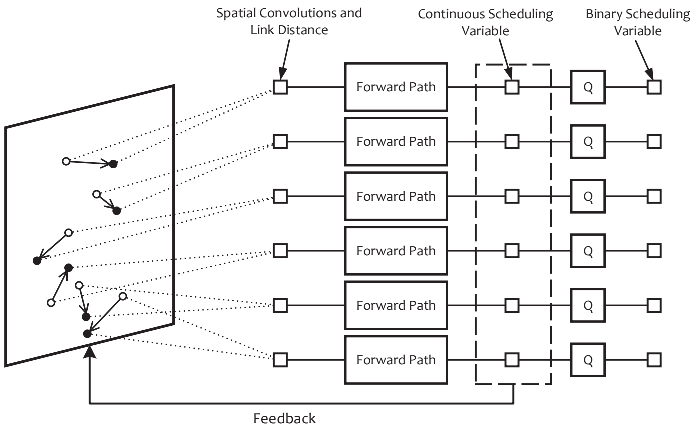
The value x_i for this link is computed based on its feature vector through the functional mapping of a fully connected neural network (denoted as F_{fc} below) with trainable weights computed through multiple layers with nonlinearities:

$$x_i \leftarrow F_{fc}(\text{TxINT}_i, \text{RxINT}_i, \|\mathbf{d}_i^{\text{tx}} - \mathbf{d}_i^{\text{rx}}\|_2). \quad (8)$$

The convolution stage and the fully connected stage together form one forward computation path for each transmitter-receiver pair, as depicted in an example in Fig. 3. The example in Fig. 3 shows a neural network with three inputs and one hidden layer. In actual implementation, we use a feature vector of size 7, consisting of three different ranges of convolutions for both the transmitter and the receiver, plus the distance information. We use two hidden layers with 21 neurons each to enhance the expressive power of the neural network. A rectified linear unit (ReLU) is used at each neuron in the hidden layers; a sigmoid nonlinearity is used at the output node to produce a value in $[0, 1]$.

3) Feedback Connection: The forward computation (which includes the convolution stage and the fully connected stage) takes the link activation pattern x_i as the input for constructing the density grid. In order to account for the progressive (de)activation pattern of the wireless links through the iterations, i.e., each subsequent interference estimates need to be aware of the fact that the deactivated links no longer produce or are subject to interference, we propose a feedback structure, in which each iteration of the neural network takes the continuous output x from the previous iteration as input, then iterates for a fixed number of iterations. We find experimentally that the network is then able to converge within a small number of iterations (e.g., with fixed 5 iterations), after breaking certain symmetry (which could result in oscillation as described in the next section).

The feedback stage is designed as following: After the completion of $(k-1)$th forward computation, the $\mathbf{x}$ vector of $[0, 1]$ values is obtained, with each entry representing the activation status for each of the N links. Then, a new forward computation is started with input density grids prepared by feeding this $\mathbf{x}$ vector into (6)-(7). In this way, the activation status for all N links are updated in the density grids for subsequent


Spatial Convolutions and Link Distance
Continuous Scheduling Variable
Binary Scheduling Variable
Forward Path
Q
Feedback
Fig. 4: Overall neural network with one forward path per link and with feedback connections and quantized output (denoted as "Q").

interference estimations. Note that the trainable weights of the spatial convolutional filter and the fully connected stage for the multiple iterations are tied together for more efficient training. This feedback structure is depicted in Fig. 4.

4) Scheduling Outputs: After a small fixed number of iterations, the scheduling decision is obtained from the neural network simply by quantizing the $\mathbf{x}$ vector from the last iteration into binary values, representing the scheduling decisions of the N links.

D. Training Process

1) Targets Preparation: The network is trained in a supervised fashion. We generate a large number of randomly located wireless links in a fixed geographic area, then use the wireless channel propagation and fading model to generate the channel realizations, and finally use the state-of-the-art FPLinQ algorithm [4] to produce the schedule that (locally) maximizes the sum rate objective. Large amount of training data that map the locations of the transmitter-receiver pairs to the optimized schedule are used as the targets for the neural network.

2) Training Setup: Using supervised learning with binary targets, the network is trained end-to-end using TensorFlow with the cross-entropy (CE) loss between the targets and the actual network outputs as the cost function. Cross-entropy is a commonly used cost function measuring the "distance" between two probability distributions p and q, defined as

$$\text{Cross\_Entropy}(p, q) = -\mathbb{E}_p[\log q]. \quad (9)$$

To allow the gradients to be back-propagated through the network, we do not discretize the network outputs when computing the CE loss. Therefore, minimizing the cost function actually encourages the neural network's continuous outputs to converge to a binary distribution.

E. Symmetry Breaking

The overall neural network is designed to encourage links to deactivate either when it produces too much interference to its neighbors, or when it experiences too much interference

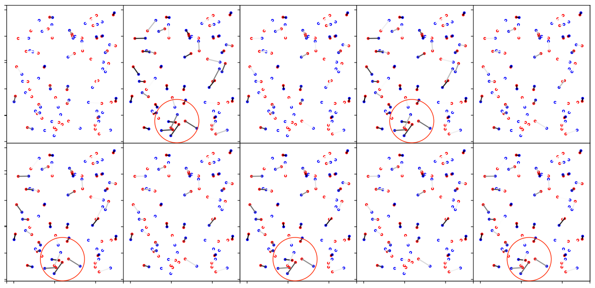


Fig. 5: Oscillatory behavior in the neural network training process.

TABLE I: Design Parameters for the Spatial Deep Neural Network

Parameters	Values	
Dimensions of the Convolution Filters	Small	5 cells × 5 cells
	Medium	15 cells × 15 cells
	Full	31 cells × 31 cells
Feature Vector	7 elements per link	
First Hidden Layer	21 units	
Second Hidden Layer	21 units	
Number of Iterations	Training	5 iterations
	Testing	10 iterations

from its neighbors. However, because training happens in stages and all the links update their activation pattern in parallel, the algorithm frequently gets into situations in which multiple links may oscillate between being activated and deactivated.

Consider the following scenario involving two closely located links with identical surroundings. Starting from the initialization stage where both links are fully activated, both links see severe interference coming from each other. Thus, at the end of the first forward path, both links would be turned off. Now assuming that there are no other strong interference in the neighborhood, then at the end of the second iteration, both links would see little interference; consequentially both would be encouraged to be turned back on. This oscillation pattern can keep going, and the training process for the neural network would never converge to a good schedule (which is that precisely one of the two links should be on). Fig. 5 shows a visualization of the phenomenon. Activation patterns produced by the actual training process are shown in successive snapshots. Notice that the three closely located strong interfering links located at middle bottom of the layout have the oscillating pattern between successive iterations. The network could not converge to an optimal scheduling where only one of the three links are scheduled. The same happens to the two links in the upper left part of the area.

To resolve this problem, this paper proposes a stochastic update mechanism to break the symmetry. At the end of each forward path, the output vector $\mathbf{x}$ contains the updated activation pattern for all the links. However, instead of feeding back $\mathbf{x}$ directly to the next iteration, we feedback the updated entries of $\mathbf{x}$ with 50% probability (and feedback the old entries of $\mathbf{x}$ with 50% probability). This symmetry breaking is used in both the training and testing phase and is observed to benefit the overall performance of the neural network.

IV. Experimental Validation

A. *Fixed Network Topology*

We use a wireless D2D scenario consisting of $N = 50$ D2D pairs randomly deployed in a 500 meters by 500 meters region to validate the proposed approach. The locations for the transmitters are generated uniformly within the region. The locations of the receivers are generated according to a uniform distribution within a pairwise distances of $2 \sim 65$ meters from their respective transmitters. The channel model is adapted from the short-range outdoor model ITU-1411 with a distance-dependent path-loss, over 5MHz bandwidth at 2.4GHz carrier frequency, and with 1.5m antenna height and 2.5dBi antenna gain. The transmit power level is 40dBm; the background noise level is -169dBm/Hz. We assume an SNR gap of 6dB to Shannon capacity formula to account for the non-ideal coding and modulation in practice.

We randomly generate many samples of the D2D network. For each specific layout and each specific channel realization, the FPLinQ algorithm [4] is used to generate the target sum-rate maximizing scheduling output. We generate 2.1 million such samples for training, and 5000 samples for validation/testing. Significant amount of tuning of the training process is involved to prevent model over-fitting. The numerical results reported here also involve using training samples generated at higher SNR than the setting mentioned above.

The design parameters for the neural network are summarized in the Table I. We compare the sum rate performance achieved by the trained neural network with each of the following benchmarks in term of the average sum rate over all the testing samples. The benchmarks are:

- **All Active:** Activate all links; treat interference as noise.
- **Random:** Schedule each link with 0.5 probability.
- **Strongest Links First:** We sort all links according to the direct channel strength, then schedule a fixed portion of the strongest links. The optimal percentage is taken as the average percentage of active links in the FP target.
- **Greedy:** We sort all links according to the direct link strength, then schedule one link at a time. We choose a link to be active only if scheduling this link strictly increases the sum rate. Note that the interference at all active links needs to be re-evaluated in each step as soon as a new link is turned on or off.
- **FP:** We run FPLinQ for 100 iterations and take the resulting output.

In the first experiment, we do not include fast fading in the channel model, and report the sum rate performance in the first column of Table II. The performance is expressed as the percentages as compared to FPLinQ. For a more thorough examination of the distributions of the sum rate, a cumulative distribution plot across all the testing samples is shown in Fig. 6. As shown in the first column of Table II, the proposed spatial learning approach achieves more than 98% of the aver-

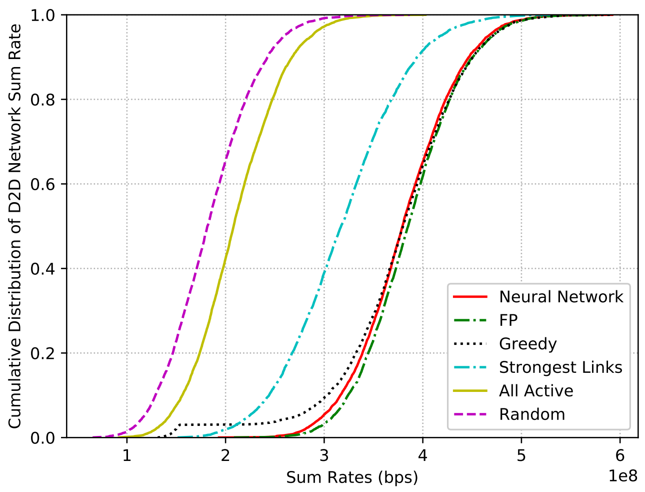


Fig. 6: Cumulative distribution of sum rate over network layouts.

TABLE II: Average Sum Rate Performance

Sum Rate (%)	CSI	No Fading	With Fading
Spatial Deep Learning	✗	98.61	88.31
Greedy	✓	97.08	98.21
Strongest Links	✓	82.03	80.80
Random	✗	47.60	44.48
All Active	✗	54.18	50.37
FP	✓	100	100

age sum rate produced by FPLinQ *without explicitly knowing the channels*. We note that in this case of without fast fading, the channel coefficients are essentially deterministic functions of the distance. Thus, this experiment demonstrates that the proposed neural network architecture is able to accurately learn this deterministic function based on the training samples. Although the neural network shows comparable results as the greedy heuristic in the average sum rate, it is worth emphasizing that the greedy heuristic utilizes the channel state information as input, thus making use of the $O(N^2)$ channel coefficients as opposed to the $O(N)$ location information.

As a second experiment, we add fast fading to the channel model. Thus, the channel coefficients are now stochastic functions of the link distance. The sum rate results of this case are summarized in the second column of Table II. The proposed spatial deep neural network without CSI still has respectable performance of 88.3% as compared to the benchmark methods (which all use exact channel coefficients). The performance loss of the neural network is due to that fast fading is not accounted for during training, but affects the achievable rate in testing. As a side comparison, if we had not used the fast fading information in FPLinQ, the FP's performance would also have dropped to about 89.0%. Thus, the 88.3% achieved by the neural network indicates that it has learned to within 1% of the best that one can hope to achieve for finding the optimal schedule without exact channel state information.

V. Conclusion

Deep learning has had remarkable success in many machine learning tasks, but the ability of deep neural networks to learn the outcome of large-scale discrete optimization in still an open research question. This paper provides evidence that for the challenging scheduling task for the wireless D2D networks, deep learning can perform very well for sum-rate maximization. In particular, this paper demonstrates that by using a novel geographic spatial convolution for estimating the density of the interfering neighbors around each link and a feedback structure for progressively adjusting the link activity patterns, a deep neural network can in effect learn the network interference topology and perform scheduling to near optimum based on the geographic spatial information alone, thereby eliminating the costly channel estimation stage.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, pp. 436–444, May 2015.

[3] K. Hornik, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, pp. 359–366, 1989.

[4] K. Shen and W. Yu, "FPLinQ: A cooperative spectrum sharing strategy for device-to-device communications," in *IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2323–2327.

[5] ——, "Fractional programming for communication systems—Part I: Power control and beamforming," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2616–2630, May 15, 2018.

[6] ——, "Fractional programming for communication systems—Part II: Uplink scheduling via matching," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2631–2644, May 15, 2018.

[7] 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.

[8] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Trans. Signal Process.*, vol. 59, no. 9, pp. 4331–4340, Apr. 2011.

[9] 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.

[10] 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, Jun. 2016.

[11] B. Zhuang, D. Guo, E. Wei, and M. L. Honig, "Scalable spectrum allocation and user association in networks with many small cells," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 2931–2942, Jul. 2017.

[12] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAN: Distributed randomized TDMA scheduling for wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 10, pp. 1384–1396, Oct. 2009.

[13] L. P. Qian and Y. J. Zhang, "S-MAPEL: Monotonic optimization for non-convex joint power control and scheduling problems," *IEEE Trans. Wireless Commun.*, vol. 9, no. 5, pp. 1708–1719, May 2010.

[14] M. Johansson and L. Xiao, "Cross-layer optimization of wireless networks using nonlinear column generation," *IEEE Trans. Wireless Commun.*, vol. 5, no. 2, pp. 435–445, Feb. 2006.

[15] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for wireless resource management," in *IEEE Int. Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, Jul. 2017, Full version [Online] Available: https://arxiv.org/abs/1705.09412.

[16] F. Liang, C. Shen, and F. Wu, "Power control for interference management via ensembling deep neural networks," 2018, preprint.

[17] J. G. D. Forney and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, Oct. 1998.