Deep Learning Meets Wireless Network Optimization: Identify Critical Links

Lu Liu, *Member, IEEE*, Bo Yin, *Student Member, IEEE*, Shuai Zhang, *Student Member, IEEE*, Xianghui Cao, *Senior Member, IEEE*, and Yu Cheng, *Senior Member, IEEE*

Abstract—With the superior capability of discovering intricate structure of large data sets, deep learning has been widely applied in various areas including wireless networking. While existing deep learning applications mainly focus on data analysis, the role it can play on fundamental research issues in wireless networks is yet to be explored. With the proliferation of wireless networking infrastructure and mobile applications, wireless network optimization has seen a tremendous increase in problem size and complexity, calling for a paradigm for efficient computation. This paper presents a pioneering study on how to exploit deep learning for significant performance gain in wireless network optimization. Analysis on the flow constrained optimization problems suggests the possibility that a smaller-sized problem can be solved while sharing equally optimal solutions with the original problem, by excluding the potentially unused links from the problem formulation. To this end, we design a deep learning framework to find the latent relationship between flow information and link usage by learning from past computation experience. Numerical results demonstrate that the proposed method is capable of identifying critical links and can reduce computation cost by up to 50 percent without affecting optimality, thus greatly improve the efficiency of solving network optimization problems.

Index Terms—Optimization, machine learning, deep learning

1 Introduction

Machine learning, an artificial intelligence technology that learns patterns from empirical data, has found its applications in classification, regression, prediction and control. Fueled by the exploding data complexity, and enabled by the advancements in computing hardware, deep learning, a branch of machine learning, has received recognition in both academia and industry for its potential for efficiently extracting intricate structures in large data sets [1]. Deep learning techniques have been reported to assist detection or classification in the wireless networking scenarios [2], [3], [4], [5]. However most of the existing attempts only apply deep learning as an auxiliary tool, detecting features in scenarios similar to those in computer vision or recommendation system; it has yet to be seen how deep learning would fundamentally impact performance optimization in wireless networks.

Modern wireless networks are tremendously more complex than before and the traditional approach to resource allocation problems by optimization theory has become more tenuous than ever. Today's networks are designed to support a large and diverse user base, with the deployment and interplay of multi-layer and heterogeneous structures. The optimal allocation and scheduling of these various network resources raises a critical concern in the computational efficiency [6], [7]. Therefore, developing efficient algorithms to solve large-scale optimization problems becomes an emerging task for future networks. It could be observed that a part of the computing efforts is "wasted" due to repeatedly solving for similar conditions. For example, in a flow scheduling problem under a given topology, a slight change in flow demand or location may not lead to significant change in the results, yet a new optimization problem has to be formulated and the expensive iterative computations repeated. It is more efficient if the results are "memorized" somehow such that the historical information can contribute to solving future problem instances with similar conditions. In other words, through solving these problems and digesting the solutions, the system could gain "experience", which as an example could be the structural features of the network or agent distribution, to facilitate future decision-making. Considering its recent success, deep learning could be a key technology in carrying out such a task.

In the existing works dealing with the large scale optimization problems, decomposition and distributed solution are the main approaches. Delayed column generation (DCG) [6], [8], [9] has been exploited as a decomposition technique in solving optimization problems. However, in these works, DCG is not able to reduce the number of constraints, leading to considerable time consumption in solving the decomposed problems. Game theoretical distributed approach is suitable for solving the problems where a large number of users/players share or compete for network resources [10]. Since the players take actions in parallel and in a distributed manner where each one only needs to consider a small scale problem locally, the original problem can be solved efficiently. But due to the price of anarchy, it

- *L. Liu, B. Yin, S. Zhang, and Y. Cheng are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616. E-mail: {lliu41, byin, szhang104}@hawk.iit.edu, cheng@iit.edu.*
- *X. Cao is with the School of Automation, Southeast University, Nanjing 210096, China. E-mail: xh.cao@ieee.org.*

Manuscript received 15 Oct. 2017; revised 25 Feb. 2018; accepted 2 Apr. 2018. Date of publication 17 Apr. 2018; date of current version 5 Mar. 2020. (Corresponding author: Lu Liu.)
Recommended for acceptance by W. Zhang.
Digital Object Identifier no. 10.1109/TNSE.2018.2827997

usually leads to incentive compatible but not socially efficient solutions. On the other hand, distributed algorithms such as reinforced learning [11] and dual decomposition [12] are able to provide optimal solution, but a common issue faced by these approaches is the convergence time. In order to converge to the optimal solution or to satisfy a predefined performance requirement, it usually takes long time of iteration or a large number of rounds, especially for the problems with large scale. Moreover, even when the original problem is decomposed to sub-problems or distributed for parallel computations, the resulting sub-problem will still be of large size if the number of variables or elements involved in the problem is large, which is the usual case in large scale optimization problems. That is to say, if the number of elements in the original problem is not changed, it will finally contribute to an increased computation cost either in computational complexity or computation time.

Motivated by this, we seek a novel solution procedure by reducing the problem size itself, i.e., formulating an optimization problem with smaller size while solving it yields the same solution as that of the original. By analyzing the solutions of flow-constrained optimization problems in [6], we observe that many links are not involved in the optimal solution. This observation indicates the possibility to cut off part of the elements (in this case, links) to reduce problem size without changing the solution. Then the goal is to identify useful links before solving the problem. A similar approach for link evaluation is discussed in [13] and [14], but the methods proposed in these works are based on heuristics and cannot be well adapted to changing flow demands. Therefore, the target is to design an adaptive method that is able to identify critical links under different flow input. To this end, we adopt deep learning to perform link evaluation, which is capable of extracting highly abstracted features from input and finding complicated latent relationships between input (flow demand) and output (link values) [1]. As mentioned previously, the learning framework is designed to learn from computation experiences and extract structural information from the network, which is then utilized to identify critical links in the optimization problems.

In this paper, we develop the learning based approach on a typical optimization problem in wireless networks, which is demand constrained energy minimization problem in generic multi-dimensional networks. Results from our previous work [6], [8], [9] indicate that it may not be necessary to include all the links into formulation since some of them will not be used in the optimal solution. With training data generated from off-line pre-computations of sample problems, we adopt deep learning algorithm to find the relationship between multi-commodity flow demand information and link usage in optimal solution. Since the learning task in our problem largely differs from those in the existing works, a learning framework that is suitable for link evaluation is designed. Based on the learning result, we then identify the link criticalness under different flow input, and links that are not likely to be scheduled will be excluded from the problem formulation to reduce problem size. Numerical results demonstrate that the proposed method is able to reduce the computation time to at most 50 percent compared with that of the original problem, without degrading the optimality of solution.

The idea of applying learning algorithm for link evaluation and computation reduction was first introduced in our previous work [15], where some preliminary but limited results are presented to demonstrate the validity of this novel approach. In this paper, with considerations of differences between our learning task and the typical classification tasks, we re-design the learning structure so that the unique features in the link evaluation tasks can be accommodated, including revised input and output layers and the cost function in training. A complete algorithm is provided in this paper to cover the generation of training set, selection of threshold, and how to obtain the final feasible solution. Further, different from [15], the training set and testing set in this paper are generated separately without overlapping to ensure integrity of the test, and more learning results are presented and analyzed to demonstrate different aspects of the learning scheme.

Our contributions are summarized as follows:

1) We propose a novel approach of exploiting machine learning in wireless network optimization which identifies structural level information such as link criticalness from computation experiences and the learned information is utilized to improve computational efficiency;
2) Through the analysis on a class of wireless network optimization problems, we identified parts of the solution space are not commonly represented in the solutions, such as unscheduled links in the flow constrained optimization problem, suggesting the possibility that a smaller-sized problem can be solved while sharing equal optimal solutions with the original, by excluding the potentially unused links from the problem formulation;
3) We exploit deep learning to investigate the latent relationship between flow demand and link usage by designing a learning framework that is suitable to address the unique issues in link evaluation, with which a learning based algorithm is proposed to identify critical links under different flow demands and reduce optimization problem scale for wireless networks;
4) Through numerical results, we demonstrate that the proposed learning based method is valid in estimating link criticalness, based on which the computational cost in solving optimization problems can be reduced by up to 50 percent while optimality of solution remains intact, thus greatly increases the efficiency of solving optimization problems in wireless networks.

The remainder of this paper is organized as follows. More related works are surveyed in Section 2. Section 3 describes the system model and formulation of the optimization problem, as well as the analysis on problem size. Section 4 introduces the learning framework for link evaluation and the proposed learning based algorithm for problem size reduction. Numerical results and performance evaluations are presented in Section 5. Finally, Section 6 gives the conclusion remarks.

2 Related Work

Good at discovering intricate structures in high-dimensional models, deep learning methods are demonstrating great success on tasks such as image classification, speech recognition, and language translation [1]. The key advantage of deep learning is the ability to learn complex functions from raw data, e.g., pixel values of an image, which relaxes the requirement of good feature extractors, conventionally designed by hand, and thus boosts its adoption in many domains of science [1]. For example, the work in [16] trained a deep neural network (DNN) for prospective predictions in quantitative structure-activity relationships (QSAR) and showed the superior performance of the DNN generalizes well to large diverse QSAR data sets. The work in [17] also employed the DNN to characterize the high level representations of mouse RNA-Seq data with respect to the splicing patterns of tissues.

Recent years have also witnessed an increasing adoption of machine learning techniques in the network optimization with respect to various tasks, such as dynamic optimal channel assignment approximation in cellular telephone systems [18], localization in wireless sensor networks [19], and reducing the number of unsatisfied cellular users [20]. With the advent of deep learning, data-driven approach is becoming a promising attempt to explore the optimal network resource allocation. The work in [21] analyzed the spatio-temporal structure of cellular user traffic load and discovered the existence of temporal autocorrelation and spatial correlation between neighboring base stations, which were then modeled by leveraging the state-of-the-art deep learning techniques, i.e., autoencoder and Long Short-Term Memory units (LSTMs), respectively. The resultant spatio-temporal model improved the prediction accuracy in terms of traffic load, making the resource allocation more flexible. As in the other fields, the existing applications of learning in networking largely rely on the features of data and target on the processing of data itself. It is not yet fully exploited that whether the powerful learning tools can benefit wireless networking in a fundamental way, such as to identify core information implied in the network structure.

Efforts to investigate the optimization problems in wireless network resource allocation, especially channel assignment, scheduling and power control, have been under way for a long time [22], [23]. Scheduling schemes are proposed in [24] and [25] to balance network performance with energy consumption in power-constrained sensor networks. Arising naturally in many wireless network resource allocation problems, the wireless network utility maximization (WNUM) model has been studied intensively. Thanks to algorithmic and technological advances, a series of methods for WNUM problems have been proposed in the last decade [26]. Among those solution methodologies, distributed algorithms are particularly attractive due to its capability to address large-scale network problems. The idea of exploiting the decomposability structure of WNUM problems, either in the primal or dual problems, has stimulated a large number of research work, which result in various of distributed algorithms targeting on different network resource allocation problems [27]. With superior convergence performance, the alternating direction method of multipliers (ADMM) has been recognized as an efficient algorithm for solving large-scale machine learning problems [28] and sparked a tremendous attention from networking community recently [29], [30], [31], [32]. In addition, game theory provides another distributed control paradigm in wireless network, especially in large wireless ad hoc network (WANET), which is summarized in [33]. As the problem scale increases along with the development of wireless networks, an inevitable issue in most of these approaches is the trade-off between convergence time and algorithm performance in terms of optimality. It is promising to design algorithms to reduce problem scale without degrading the performance of solution by identifying critical network elements, which is to be investigated in this paper.

3 Problem Formulation and Analysis

Our previous work on wireless network optimization identified a general abstraction that the resource allocation can be viewed as an independent set or transmission pattern based scheduling [6] problem, where in both cases transmission links are the core subjects in the problems. Therefore, an effective approach of improving the algorithm efficiency of network optimization is to evaluate the usefulness of links out of the optimization framework, where learning technique can be applied to discover such information based on computation experiences. To this end, deep learning will be exploited to extract structural level information from the network and identify critical links, and the learned knowledge will then be utilized to improve the efficiency of network optimization. In this section, we will elaborate the system model and formulation of general resource allocation problem as a transmission pattern based optimization problem, and discuss the feasibility of learning link values for computational complexity reduction.

3.1 System Model

3.1.1 Network Model

Consider a generic wireless network with the set of nodes $\mathcal{N}$. Each node can be equipped with one or multiple radio interfaces, with the set of all radios denoted as $\mathcal{R}$. The set of available channels in the network is defined as $\mathcal{C}$. A node can set up a transmission link to its neighbors, i.e., nodes within its transmission range. Denote all the links in the network as $\mathcal{L}$. An optimization problem in wireless network may involve various aspects and resources, such as link scheduling, throughput maximization, power control, channel assignment, etc. In this paper, we will take the optimization problem presented in [34] as an example,[1] where the objective is to minimize total energy consumption in the network while guaranteeing the flow demands of multiple commodities can be satisfied. Each commodity flow is specified by the corresponding source and destination nodes as well as the flow demand. The optimization is done by alternately activating sets of links to fulfill multi-hop transmissions from sources to destinations, accompanied with the allocation of other resources.

In wireless network optimization problems, a common cause of large problem scale is the joint optimization over multiple dimensions of resource space, which is constructed by different types of resources. For example, most wireless

1. The proposed approach can be generalized to different types of optimization problems, as will be discussed in Section 4.5.

networks can be abstracted as a multi-radio multi-channel network [6], where each node is equipped with one or multiple radio interfaces that can exploit one or multiple non-overlapping channels for transmission. In such a network, a comprehensive allocation of all types of resources is required to obtain the optimal solution, which means the coupled issues of link scheduling, routing, power control, radio and channel assignment should be jointly optimized.

The multi-dimensional modeling technique in [34] presented a way to decouple different resource dimensions. Particularly, the concept of tuple-link maps each physical link to several tuple-links, where each tuple-link is a transmission link specified by the transmitter radio, receiver radio and the utilized channel. Further, a transmission pattern is defined as a possible power allocation on all the tuple-links in the network. With these techniques, the joint optimization problem is transformed to a transmission pattern based scheduling problem, which facilitates LP formulation.

3.1.2 Interference Model

We adopt a physical interference model to characterize the interference relationships among links. Simultaneous transmissions of multiple links (or tuple-links if in a multi-dimensional model) may incur different types of interference or conflict to each other. If two links sharing the same radios are scheduled for transmission at the same time, then neither of them can work due to radio conflict. For two links with no radio conflict but working in the same channel, co-channel interference will be incurred, which affects the SINR at receiver side. In addition, if two links with no radio conflict are working in different channels, there will not be any mutual influence. Based on the above definitions, we can obtain the SINR at the receiver of link i as

$$\mathrm{SINR}_i = \begin{cases} 0, & \text{if } i \text{ has radio conflict} \\ & \text{with other active links;} \\ \frac{g_i p_i}{\sum_{j \in I_i} g_{ji} p_j + \sigma^2}, & \text{otherwise,} \end{cases}$$

where g_i is the channel gain of link i, g_{ji} is the interference coefficient from link j to link i, p_i is the transmit power of link i, I_i is the set of links working on the same channel as that of link i, and σ^2 is the random noise. Further, according to the Shannon-Hartley equation, the achievable transmission rate of link i can be expressed as

$$r_i = \mathrm{BW} \log_2(1 + \mathrm{SINR}_i), \tag{1}$$

where BW denotes the bandwidth.

With the above modeling, it can be seen that if the status (on or off) of all the links are given, the transmission rate of links as well as the total power consumption in the network can be obtained. Based on this, we define a possible status of all the tuple-links in the network as a transmission pattern, and the optimization problem can be transformed to a scheduling problem over all the patterns (denoted as $\mathcal{A}$). The solution of this scheduling problem can also imply routing and radio/channel assignment, which means we are able to obtain the desired joint solution.

3.2 Problem Formulation

The pattern based scheduling problem is to assign active time t_a for each pattern $a \in \mathcal{A}$, with the total scheduling time bounded by one normalized slot

$$\sum_{a \in \mathcal{A}} t_a \leq 1. \tag{2}$$

To minimize the total energy consumption in the network, the objective function will be formulated by summing up energy consumption of all the patterns, which is

$$\min \quad E = \sum_{a \in \mathcal{A}} P_a t_a. \tag{3}$$

where P_a is the total power consumption of pattern a, which can be obtained by $P_a = \sum_{i \in \mathcal{L}} p_{i,a}$. $p_{i,a}$ is the transmit power of link i in pattern a, which is 0 if i is in off state and p_i otherwise.

Denote the flow of commodity c on link i as $f_{i,c}$, and the set of all commodities as $\mathcal{C}$. Since the sum flow of all commodities on a link is limited by the link capacity, which is determined by the transmission rate and transmission time. Based on this, we can obtain the following constraint

$$f_i = \sum_{c \in \mathcal{C}} f_{i,c} \leq \sum_{a \in \mathcal{A}} r_{i,a} t_a, \tag{4}$$

where $r_{i,a}$ is the transmission rate of i in pattern a.

In addition, considering the flow balance, we have the following constraint for each internal node n

$$\sum_{i \in \mathcal{L}_{n+}} f_{i,c} = \sum_{j \in \mathcal{L}_{n-}} f_{j,c}, \quad \forall c \in \mathcal{C}, \forall n \in \mathcal{N} \setminus \{n_c^s, n_c^d\}, \tag{5}$$

where $\mathcal{L}_{n+}$ denotes the set of incoming links to node n and $\mathcal{L}_{n-}$ denotes the set of outgoing links from n.

Similarly, at a source or destination node, we have

$$\sum_{i \in \mathcal{L}_{n_c^s-}} f_{i,c} = \sum_{j \in \mathcal{L}_{n_c^d+}} f_{j,c} \geq d_c, \quad \forall c \in \mathcal{C}, \tag{6}$$

where d_c is the flow demand of commodity c.

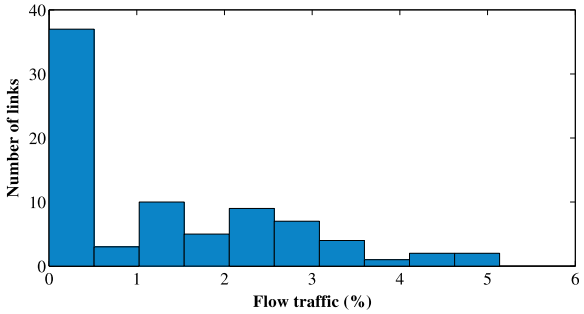
Therefore the optimization problem will be formulated as

$$\begin{aligned} \min_{\{f_{i,c}, t_a\}} \quad & \text{Objective (3)} \\ s.t. \quad & \text{Constraint (2),(4),(5)} \\ & f_{i,c} \geq 0, \quad \forall i \in \mathcal{L}, \forall c \in \mathcal{C} \end{aligned} \tag{7}$$

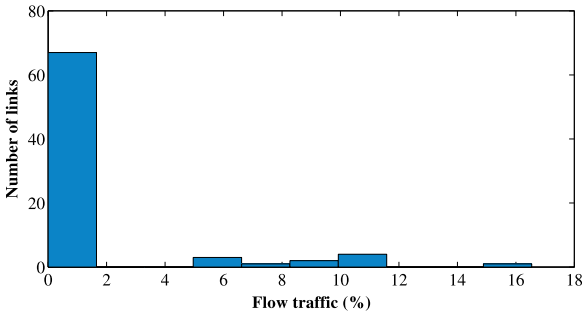
$$t_a \geq 0, \quad \forall a \in \mathcal{A}. \tag{8}$$

3.3 Analysis on Problem Scale and Solution

While the modeling in [34] successfully decoupled multiple dimensions of resources and facilitated LP formulation, another cost is introduced, which is the exponentially growing problem scale. As the scheduling object is changed from link to tuple-link, and further to transmission pattern, the solution space grows to a size with exponent equal to the number of tuple-links. For example, suppose all nodes have the same number of radios $|\mathcal{R}_0|$, the number of channels and transmit power levels are $|\mathcal{B}|$ and $|\mathcal{P}|$, respectively.



(a) Sample 1 (grid topology).



(b) Sample 2 (random topology).

Fig. 1. Histogram of flow traffic (percentage of the total amount of traffic in the optimal solution).

Then the size of the solution space is roughly $|\mathcal{P}|^{(|\mathcal{L}|\cdot|\mathcal{R}_0|^2\cdot|\mathcal{B}|)}$, where $|\mathcal{L}_p|$ is the number of physical links in the network and is in the order of $|\mathcal{N}^2|$. It is computationally impractical to find all the patterns in the optimization problem due to the exponentially large size, therefore we follow the solution proposed in [34] to solve the formulated problem, which is based on delay column generation that works on a selected subset of patterns instead of the entire set. With DCG method, although there is no need to find all the patterns, it is still time-consuming in selecting patterns, which is one of the major contributions of computation time.

The computation time in solving the optimization problem also depends on the size of the optimization problem, which is related to the dimension of optimization variables and the number of constraints. In this case, it is mainly determined by the number of tuple-links and the number of transmission patterns, while both of them are determined by the number of links $|\mathcal{L}|$. From a more general point of view, transmission link is an essential element in network optimization problems. In a generic scheduling and resource allocation problem, the solution is indeed to select links to activate and allocate transmission time to the activated links, as is done in our previous work [34]. With further analysis on the results, it can be shown that the number of links has a great impact on the computational cost of the solution, which affects both the convergence time and subproblem scale. Therefore, reducing the number of links involved in the formulation will be an efficient way to reduce the complexity.[2]

2. The reason that we are targeting at reducing the number of links instead of that of tuple-links is, tuple-links corresponding to the same link are equivalent in their criticalness in our optimization problem, which is determined by the geographical locations. Besides, the number of tuple-links is determined by that of links, therefore reducing the number of links will also reduce the number of tuple-links.

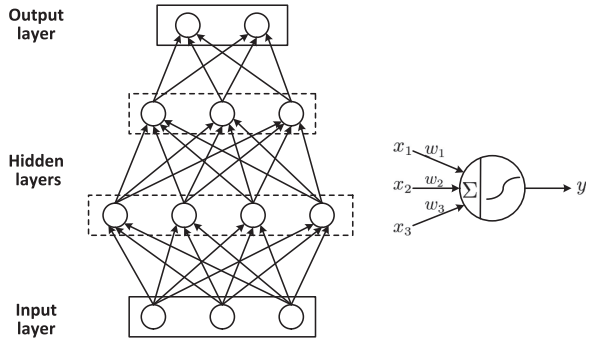


Fig. 2. Basic learning framework.

In fact, not all the links are used in the optimal solution. Fig. 1 shows two examples of link usage in the optimal solution from sample problems solved by the algorithm proposed in [34]. We calculated the amount of traffic carried on each physical link in the optimal solution and drew the histogram of flow traffic as the percentage of the total amount of traffic in the network. It can be observed that the flow demands are fulfilled by a small number of links, while the others remain idle. Whether a link will be used in the optimal solution is determined by many factors. For example, if a link is not on the paths between source and destination pairs or far away from the source or destination nodes, it will not be covered in the solution. In addition, if a link has poorer transmission quality (in terms of link capacity) compared to a nearby link, it will be unlikely to be scheduled.

Inspired by the above observation of link usage, if the unused links can be excluded from the formulation, the optimization solution can remain unchanged. This will further lead to enhanced computational efficiency for reduced computation time and storage cost. However, the previous observations on link usage are made after the solutions are obtained. In order to reduce computational complexity, we need to develop a technique to predict the link usage before solving the optimization problem.

For a fixed network topology (physical locations of nodes), the main factor that determines whether a link will be scheduled in the optimal solution is the flow information, which includes the locations of source and destination nodes and flow demand for each commodity. Then the link usage prediction problem is to perform link evaluation based on the flow information. Equivalently, the problem is to find the relationships between the input flow information and the output link evaluations.

Generally, flow information affects link usefulness in an indirect way and there is no explicit relationship between them. One approach is to leverage the knowledge obtained from solutions of sample computations, based on which we can predict the usage of links for a new problem, i.e., new input flow information. This motivates us to exploit machine learning techniques to predict link usage, which is to be elaborated in the next section.

4 Deep Learning Based Link Evaluation

In Section 3.3 we have shown that the link usage are partly determined by the quality and node locations with respect to the flow path, and because of this, intermediate characterizations may be extracted to link the flow information to

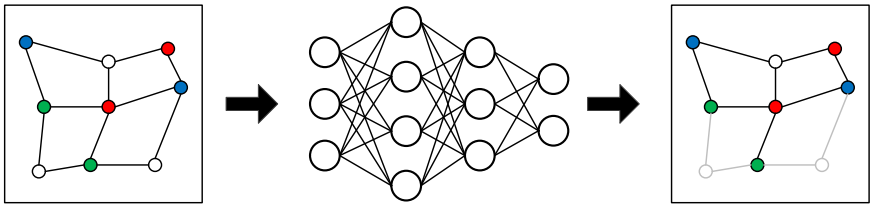


Fig. 3. Algorithm illustration. Given the topology, source/destination nodes and flow demands of multiple commodities in the network, the learning framework is supposed to automatically find which links are more likely to be used in the optimal resource allocation solution.

link usage. To give an intuitive example, given node locations and a flow demand with both source and destination nodes located in the upper part of the network, then links in the upper half of the network are more likely to be used than the lower part. The information "upper half of network should be given more consideration" can be derived through problem inputs, and such intermediate information, or *features* are what we hope to extract. In engineering practice such heuristic approaches are widely used for manual optimization, however with learning techniques it is possible to discover them in a systematic manner. Among the various existing learning frameworks, many of them are built with classification tasks in mind and feature selection and optimization take up a significant portion of effort, which are undesirable in our scenario where these unknown combination of features may make a difference in scheduling performance. To address this, we leverage deep learning framework to perform such tasks. An illustration of the algorithm is shown in Fig. 3.

4.1 Basic Learning Framework

Fig. 2 shows a general structure of deep learning framework, where the input layer represents the flow information, and the output layer represents the link evaluations. The input and output layers are connected by multiple hidden layers through internal adjustable parameters, i.e., the weights, and the value of each unit in each layer (except for the input layer, whose values are given) is determined by the units in its previous (lower) layer. In the commonly used supervised learning model, we first collect a set of input and output (label) value pairs as the training set. When an input is fed to the learning network, it produces an output which may differ from the desired value (label) in the training set. Then the weights are adjusted to minimize the difference, which is characterized by a cost function. The adjustment can be done with the back propagation (BP) method which modifies the weights layer by layer backward. The learning network is trained as the weights are adjusted according to all the training samples. Then when a new input, the test sample, is applied, the learning framework will produce an output which is an estimation on the true value. Then the difference between the learning network's output and the true output can be used to evaluate the performance of the learning network.

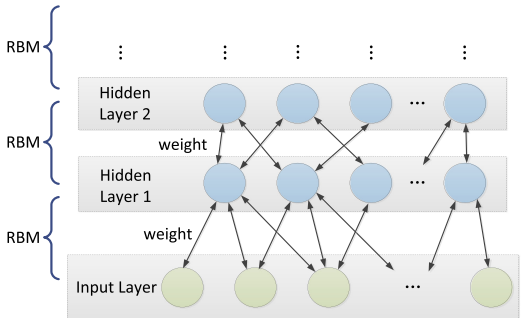


Fig. 4. DBN structure (unsupervised): stacking of RBMs.

4.2 Deep Belief Net

Generally the weights in learning networks are randomly initialized. In deep learning framework, it would be beneficial to initialize them to sensible values through pre-training, which is to apply unsupervised learning on the lower layers of the network [1]. To this end, we exploit deep belief net (DBN) [35] method to build the learning framework in our link evaluation problem.[3]

DBN consists of multiple stacks of restricted Boltzmann machine (RBM), which is a generative network with only one stochastic hidden layer (single layer). In RBM, there is no connections between the units within the same layer, and the units in the hidden layer are connected to the input layer by undirected and symmetric weights. The units in both layers are binary-valued, while the probability of turning on a unit h_i in the hidden layer is determined by

$$p(h_i = 1) = \frac{1}{1 + \exp(-b_i + \Sigma_j x_j w_{ji})}, \tag{9}$$

which is a sigmoid function of the weighted sum of the input values x_j with a bias b_i.

Since the weights are all symmetric, in addition to inferring the states of hidden layer units from the input layer, RBM is also able to reconstruct the input from the hidden layer. The training of RBM is then to minimize the reconstruction error by adjusting the weights and repeating the inferring and construction.

A single layer of RBM is not enough to fulfill the task of extracting complicated and intricate information from the input (such as the task of predicting link usage based on input flow information in this paper). Therefore, multiple RBMs are stacked to form a deep structure, the DBN [35]. In DBN, each RBM takes the output from the previous level of RBM as input, as shown in Fig. 4.

With this structure, the top layer of DBN is able to extract abstracted features from the input data, and the level of abstractions is determined by the number of layers. Although multiple layers are combined in DBN, the training

3. Generally, there is no limitation on what learning algorithm should be used in the proposed link identification task. DBN is chosen for not relying on specific input data structure and good generalization ability.

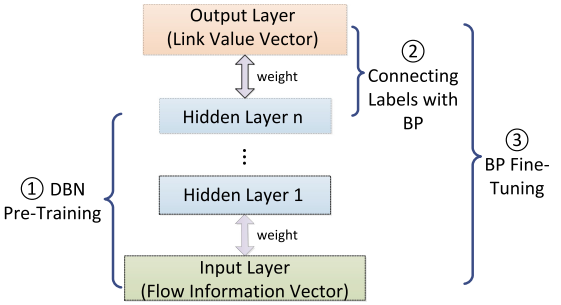


Fig. 5. DBN with BP fine-tuning (supervised).

can be done greedily by training each layer of RBM at a time [35].

DBN can be used as an unsupervised learning model, since it does not rely on output labels of training samples. However, it can also be used for supervised learning by adding the layer of labels on top of the trained DBN. In order to match the top layer of DBN with the real output layer of labels, BP method can be applied to obtain the connecting weights, and then applied to all the weights in the learning framework for fine-tuning, as shown in Fig. 5. The fine-tuning will adjust the parameters to match each input sample to its corresponding label by minimizing the cost function.

With the DBN pre-training, the extracted features can be used for better discriminating different inputs and mapping inputs to the corresponding output labels. In fact, the DBN based supervised learning framework can be viewed as a traditional deep learning network with BP method, while the initial values of parameters are carefully assigned with pre-training to improve the learning performance.

4.3 Learning Framework Design

4.3.1 Input Layer

Since the learning task is to infer link usage based on the input flow information, the input layer should be designed to fully express the information of multiple commodity flows, including the locations of source and destination nodes, along with the flow demand for each flow. To this end, we define a $|\mathcal{N}| \times |\mathcal{N}|$ *flow information matrix* $\mathbf{M}$, whose entries are assigned as

$$\mathbf{M}_{ij} = \begin{cases} d_c & \text{if } (i,j) \text{ are the (source, destination) nodes} \\ & \text{of flow } c \text{ with demand } d_c \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

The flow information matrix is then reshaped to a $1 \times |\mathcal{N}|^2$ *flow information vector*[4] $\mathbf{x}$, which is to be applied to the learning framework as the input layer.

In order to be consistent with the property of probability and sigmoid function, the input layer of DBN only accepts values between 0 and 1, which means a normalization needs to be done on the input vector that shrinks the range of input values to the interval $[0, 1]$. Since the minimum value of $\mathbf{x}$ is 0, the normalization can be done by dividing the original values with the maximum value in the input vector. To avoid bias among cases, the maximum value should not vary with different input, therefore we take the global maximum value of flow demand $d_{\max}$ as the normalizer, which is the maximal demand that can be set to a flow. In practice, $d_{\max}$ is usually predefined in the network optimization problem in order to obtain feasible solutions since the network capacity is bounded. In summary, the normalized flow information vector will act as the input layer, which is

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{d_{\max}}. \tag{11}$$

4.3.2 Output Layer - Training Samples

The target of the learning networks is to predict the usage or usefulness of each link in the network given the input and topology. Therefore the output layer can be designed as a vector with length equal to the number of links in the network.

For the output layer of a training sample, the link value should be defined based on the actual usage, which can be obtained by calculating the amount of traffic it carries in the optimal solution, as done in Section 3.3. It should be noticed that a link with more traffic than others is not necessarily more "useful". In this sense, if we directly take the traffic amount to represent the link value, some actually used links with light traffic might be overwhelmed by those with large values. In order to avoid such situation and cover all the links present in optimal solution, we use binary values to define the link value vector, which is

$$\mathbf{y}_i = \begin{cases} 1 & \text{if link } i \text{ is used in the optimal solution} \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

The binary definition indicates that we focus more on whether a link will be involved in the optimal solution, rather than how often it will be used, since the former is critical in deciding the size of problem formulation.

For a general learning based classification model, the output vector represents the label corresponding to the input data. If there are K categories, the output vector will be a K-dimensional vector with only one element of 1 representing the correct class and all the other elements zero. However, our problem is not a classification problem, which can be seen from both the physical meaning and the output vector, which may contain multiple 1's.

4.3.3 Output Layer - Test Samples

While the link values for training samples are known to us and thus defined as binary, the link values of test samples, which is the output of the learning framework and prediction of whether a link will be used, will be continuous values within the interval $[0, 1]$.

This above mentioned difference with classification model is also reflected in the output with test samples. Recall that in DBN, the value of each unit falls between 0 and 1, so does the output values. In classification problem, the output is usually transformed to probability distribution by applying normalization or softmax function, and the transformed values indicate the possibilities that the test sample belongs to each class, which can be interpreted as the confidence that the classifier has in categorizing the input into each class. Similarly,

4. The entries on diagonal can be removed since they are always zero and carrying no information. Then the corresponding vector will be of size $1 \times |\mathcal{N}|(|\mathcal{N}| - 1)$.

in our problem, the output can be used to express the probability of whether a link will be used. However, different from classification problems whose output values sum to 1, in our model, the sum may exceed 1 since multiple links may be considered useful. Therefore normalization is not applied in our model, and the raw output is taken as evaluation of links.[5]

4.3.4 Training Set

Having defined the input and output layers, the training set can be built as pairs of flow information vectors and binary link values obtained from optimal solutions. The training data can be gathered from either the historic solutions or offline solving sample problems, and can be further augmented with newly solved problems. In our case, we generate a large number of sample problems to build the training set. Particularly, each training sample is obtained by solving the optimization problem formulated with a randomly generated flow information matrix (vector), and recording the links that are used in optimal solution as the link value vector. The training process is summarized in Algorithm 1.

Algorithm 1. Generating Training Set

Initialization: $i = 0$;
while $i <$ training set size **do**
  $i = i + 1$;
  Randomly generate a set of flow commodities and demands, convert to flow information vector $\mathbf{x}$;
  Formulate the optimization problem with the flow information;
  Solve the optimization problem with method in [34];
  Record the link used in the optimal solution, calculate output link value vector, denoted as $\mathbf{y}$;
  Add input/output pair $(\mathbf{x}, \mathbf{y})$ to the training set;
end

4.3.5 Cost Function

While we still use cross entropy as the cost function, since the output values are no longer a probability distribution, we modify the cost function as

$$-\frac{1}{S}\sum_{i=1}^{S}\sum_{i=1}^{|\mathcal{L}|}[y\ln\hat{y} + (1-y)\ln(1-\hat{y})], \tag{13}$$

where S is the batch size in training, y stands for the true value of a link and $\hat{y}$ is the output value of the learning framework. Then the parameter of the learning framework will be adjusted with conjugate gradient method layer by layer to minimize the cost function in a back prorogation manner, which starts from the top layer and goes through the bottom layer.

It is worth noted that the value of S will have an effect on the learning performance. The target of making training samples into batches instead of dealing with the entire training set is to reduce the computational complexity. In a classification problem, the batch size is set so that roughly each batch contains samples from all classes to avoid bias. Therefore the size of a batch can be set to equal or larger than the number of classes. In our problem, if the number of links is viewed as the number of classes, the batch size should be set to the number of links (or slightly larger).

4.3.6 Training

The training of the learning framework is summarized in Algorithm 2. First, the training set is built with Algorithm 1. Then the DBN is trained as multiple layers of RBMs with unsupervised training [35]. The output (top) layer of the pre-trained DBN is then connected with true link values with randomly initialized weights for supervised training. BP training is first performed only on the added layer of weights while the weights within DBN are fixed for several rounds, then the training is further performed on all the layers for fine-tuning [35]. The training process will assign values for all the interacting weights in the learning structure.

Algorithm 2. Training of Deep Learning Framework

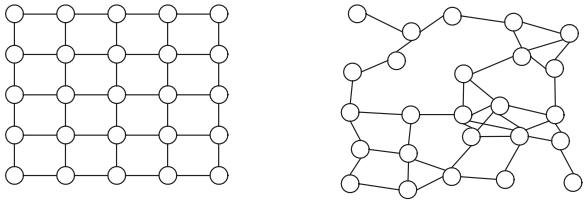
1. Build training set with Algorithm 1;
2. Train DBN with unsupervised training (Step 1 in Fig. 5);
3. Add output layer of true link values, connect the output layer to the pre-trained DBN with random weights;
4. Hold weights in DBN while adjust weights in the top layer for a specified number of rounds with BP method (Step 2 in Fig. 5);
5. Fine-tune the weights in all the layers with BP (Step 3 in Fig. 5);

4.4 Algorithm Design

With the training results, we may input the new flow demand vector from the problem we are going to solve. Then the output of the learning framework will provide the evaluation vector on all the links. The result can be viewed as an estimation on whether a link would be used under the new flow demand vector, indicated by the values in the vector. Based on the evaluation, we can keep the links with higher values and exclude those with very small values, which is done by applying a threshold.

The threshold is defined based on the output link values. Both the absolute values and the relative values of links can be used as a reference for defining a proper threshold. The relative values can be used to define a threshold that can directly control how many links will be omitted. For example, if we want to cut off 20 percent of the links, then the threshold can be set to keep link with the top 80 percent values. However, it may be difficult to define the cut-off percentage without knowing the actual percentage of links used in the optimal solution, since more links should be kept if more links are involved in the solution. Considering that the absolute value of a link can reflect its criticalness estimated by the learning framework, a constant value (e.g., 0.5) can be used as threshold to select the critical ones. But in this case, the portion of remaining links may vary largely in different problems due to different learning performance in these problems. Taken both aspects into

5. In order to be consistent with the commonly implemented learning frameworks, we first applied normalization to map the sum to 1, where we found that after normalization, most links only got very small values, even the useful ones. This made the learning framework think that none of the link is useful and the result deviating from the true output. Therefore we removed the normalization and kept the raw output.



(a) Grid topology. (b) Random topology.

Fig. 6. Topology.

consideration, we design a threshold that can reflect both the absolute and relative values of links, which is

$$y_{\text{thr}} = \alpha \bar{y}, \tag{14}$$

where $\bar{y}$ is the average link value and α is a coefficient. In this case, adjusting the value α can help control the percentage of links remained, and $\bar{y}$ provides a reference on the absolute link values specified for different cases.

Generally, the threshold can be selected according to the experiences from the training. A self-test on the training set can be conducted to check the average percentage of links required to be kept in order to cover all the critical links. Considering that the learning performance on a test sample will be worse than that on a training sample, higher percentage of links should be remained for a test sample, and the threshold can be set accordingly.

After cutting off links according to the threshold, the optimization problem will be re-formulated with only the links that are kept. Since the learning results may not be perfectly accurate, it is possible that a really critical link may be underestimated and excluded from the formulation. Then the reformulated problem may have no feasible solution. In this case, the threshold will be lowered to cut off less links and the problem is solved again. The algorithm is summarized in Algorithm 3. Generally, smaller number of links will lead to less optimization variables and constraints, which reduces the computation time and storage cost. Examples of the cost reduction will be shown in the next section.

Algorithm 3. Deep Learning Based Link Evaluation and Computation Reduction

Input: Trained learning framework (parameter values); flow information of a new problem (from test set);

1. Convert the flow information to the flow information vector (as in Section 4.3.1);
2. Apply the flow information vector to the learning framework to get the output link value vector, which is the estimation of link criticalness;
3. Apply threshold (Eq. 14) to cut off low-valued links;
4. Use remaining links to re-formulate the optimization problem;
5. Solve the reduced size problem, if infeasible, reduce threshold, go to Step 3;

4.5 Generalization of Results

The proposed approach is not limited to the formulation presented in this paper. In the scheduling and resource allocation problems over a multi-dimensional resource space, the objective function and constraints may vary among different network resources according to specific concern of different scenarios. However, the links are always critical elements that determine the formulated problem size. Therefore reducing the number of links involved in the formulation will benefit the improvement of optimization efficiency.

More generally, the optimization problems in wireless network may span to different area where links are not involved or not as the focus variables. In these cases, the proposed algorithm can also be applied by replacing link evaluations with the corresponding network elements in formulation and following similar procedures for training and implementation. For example, in addition to link, we can also evaluate other network elements such as nodes, independent sets ([9]), transmission patterns ([34]) with respect to their contributions in network scheduling or resource allocation. Moreover, structural level information may also include abstracted knowledge such as which area is more likely to be congested, which nodes' energy is likely to be depleted first. Through learning algorithms, the extracted information can be utilized to improve the efficiency of optimization or directly enhance network performance.

5 Performance Evaluation

In this section, we present numerical results of the proposed deep learning based optimization. The optimization is performed in a network with 25 nodes with a grid or random[6] deployment (as shown in Fig. 6, where each node is equipped with one or multiple radio interfaces. Transmissions can take place on one or multiple non-overlapping channels. We consider the topology of nodes fixed, while the flow information changes by each sample case. The goal is to evaluate link criticalness through off-line training, based on which the optimization problem can be solved in reduced time under new flow demand.

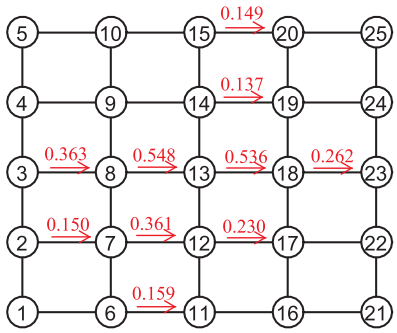
5.1 Link Evaluation

We first construct the training set by solving pre-generated optimization problems, where each case is formulated with randomly generated flow demand vector and solved with DCG-based decomposition algorithm. Solving each problem will provide one pair of input (flow information vector) and output (link value vector), and the data pairs are used as training samples for the learning algorithm. Following the same way, a test set is built to test the performance of the leaning results. The learning framework is constructed with 3 hidden layers with 500, 400, 200 units, respectively. We input the test flow vectors to the learning structure to get the learning results, which are the link value vectors estimated by the learning algorithm. The learning result is then compared with the true result obtained from solving the optimization problem.

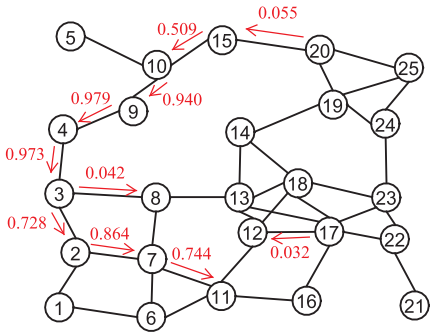
5.1.1 Single-Flow Link Evaluation

For illustration, we first test the learning algorithm with single flow scenarios, with a training set of 10000 samples. Then the test is done with a left-to-right sample flow input

6. To maintain connectivity of the network, the random topology is generated based on a grid topology, then applying both a horizontal and a vertical random offsets to each node's location.

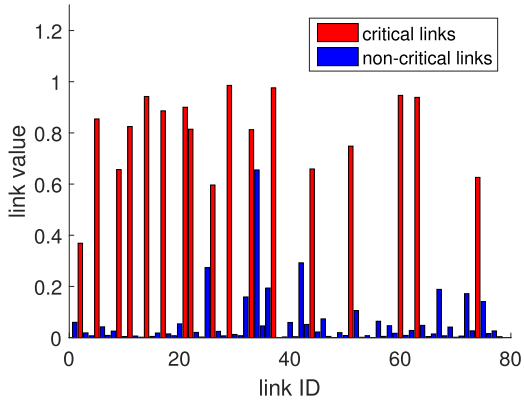


(a) Sample flow from Node 3 to Node 23. The optimal path should be $3 \rightarrow 8 \rightarrow 13 \rightarrow 18 \rightarrow 23$.

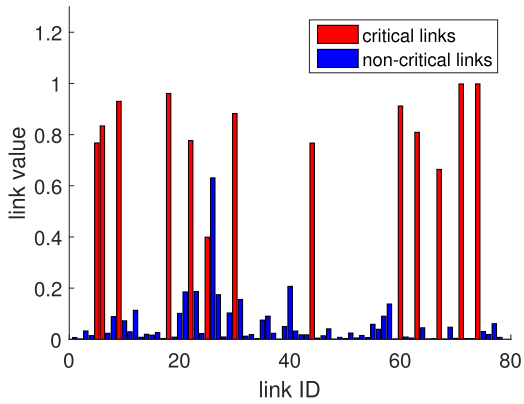


(b) Sample flow from Node 15 to Node 11. The optimal path should be $15 \rightarrow 10 \rightarrow 9 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 7 \rightarrow 11$.

Fig. 7. Single flow link evaluations.



(a) Sample 1.



(b) Sample 2.

Fig. 8. Multi-flow link evaluations.

in grid topology, and a top-to-down sample flow in random topology. The output link values are sorted in the descending order and the top 10 valued links for each case is shown in Fig. 7.

As can be seen from Fig. 7, the link evaluation from the learning algorithm correctly identified the critical links by outputting relatively larger values for these links. Especially for Sample (b), where only the links on the optimal path get values larger than 0.5 and all the other links get very small values. In Sample (a), in addition to the really critical links, some links off the optimal path (e.g., link from Node 7 to Node 12, link from Node 12 to Node 17) also get large output values. In fact, this result indicates a possible alternate choice of link due to proximity to the optimal path, and such information may be useful under multi-flow scenarios where the optimal path gets crowded.

5.1.2 *Multi-Flow Link Evaluation*

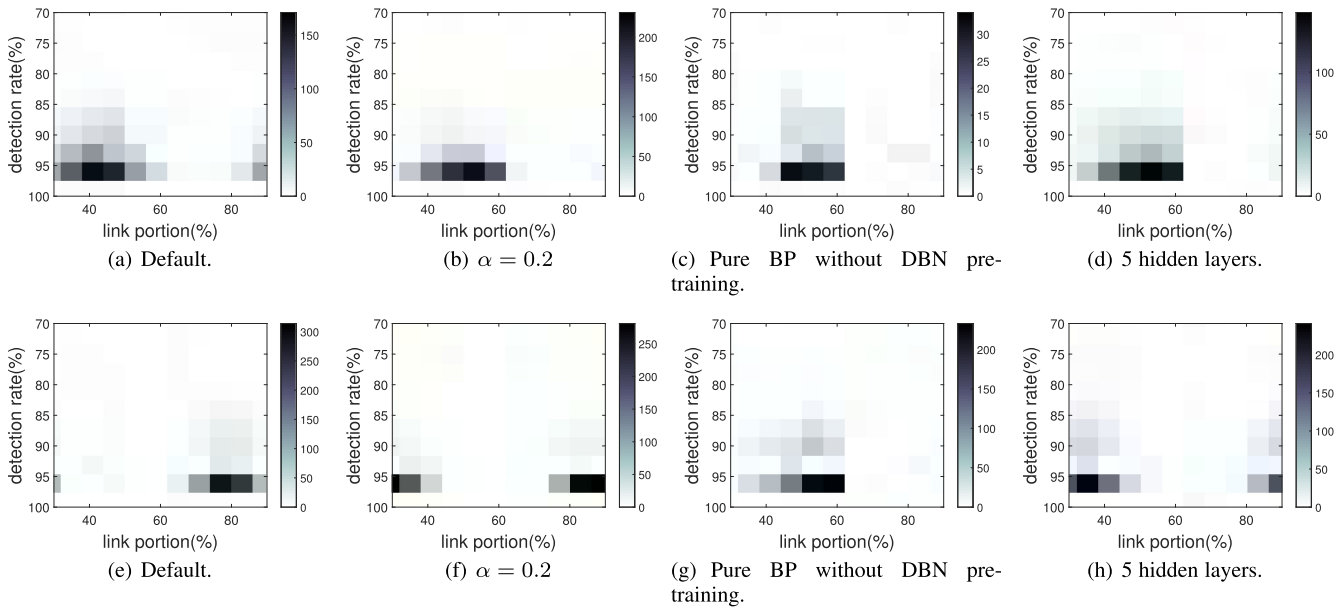
The single-flow scenario is a simple case for the learning framework in the sense that the test samples may contain an overlap with the training set since the number of possible source destination pairs is limited.[7] In the rest of this section, all the experiments are carried out in a more practical multi-flow scenario with 3 to 5 commodity flows, and non-overlapping training and testing sets with 20000 training samples and 1000 test samples. Two sample sets of link evaluation results are shown in Fig. 8. In the figures, the bar heights represent the output link values of the learning algorithm, while the bars colored red are the ones present in the optimal solution (really critical ones).

Similar to the single-flow cases, from Fig. 8 we can observe that the learning process is able to detect most of the critical links since these links generally receive higher values. Based on the evaluation results, links with higher values will be kept while lower-valued links are pruned from the formulation. With a properly selected threshold, it is possible to exclude a portion of links while still includes every critical link, which means a reduced search space size without changing the optimal result.

5.2 Learning Performance

The learning result of link evaluation will be used to determine which links will be removed and which will be kept in the formulation by applying the threshold as discussed in Section 4.4. We say a link is "detected" if it presents in the optimal solution and receives an output value higher than the threshold in the learning results, and define detection rate as the percentage of links detected (as compared with the total number of links used in the optimal solution). Fig. 9 shows the results of detection rate with remaining portion of links for all the test samples under different parameter settings. A result that is closer to the bottom-left corner has better performance, since it achieves higher detection rate with

7. When there are fewer flows, there will be fewer variety of the resulting flow distributions. Then it is more likely that the test sample can find a very similar case from the training set, resulting in good performance. On the other hand, if we have a very large training set, the performance of multiple flow scenario can also be improved. So the major factor affecting the algorithm performance is the size of training set.



(a) Default. (b) $\alpha = 0.2$ (c) Pure BP without DBN pre-training. (d) 5 hidden layers.

(e) Default. (f) $\alpha = 0.2$ (g) Pure BP without DBN pre-training. (h) 5 hidden layers.

Fig. 9. Distribution (density) of results for the 1000 sample cases in the testing set. The default setting is a learning framework of 3 hidden layers with DBN pre-training, and threshold coefficient α set to 0.3. The top four figures are for grid topology and the bottom four for random topology.

smaller link portion. Result points located at both 100 percent detection rate and link portion are omitted since they will not benefit the complexity reduction goal.

5.2.1 Detection and Reduction Trade-off

The value of threshold (as in Eq. (14)) has the most direct effect on the results. Varying the threshold, we can observe a trade-off between detection rate and the portion of remaining links, as can be seen in the comparisons between Fig. 9a and 9b, and between Fig. 9e and 9f. Generally, a higher threshold means less links kept, which further means larger computation reduction. However, it is also risky to set a high threshold since truly useful links may also be cut off. As discussed in Section 4.4, multiple concerns should be taken into consideration to set the threshold properly.

5.2.2 Effect of DBN Pre-Training

We further investigate the effects of DBN pre-training in the learning performance. The learning framework without DBN pre-training can be viewed as a pure BP neural network where all the weights are randomly initialized. The comparisons are shown in Fig. 9a, 9c and 9e, 9g. The performance of the learning with DBN pre-training is generally better than that without pre-training. The reason is that DBN pre-training can be viewed as a process to properly assign initial weight values to the multi-layer learning framework before BP method's adjustment, which takes effect by extracting features from the input that can help better map each input to the corresponding output. This advantage of DBN pre-training has already been seen in typical machine learning tasks such as classification, while the results in this paper show that it also benefits the deep learning framework in our problem.

5.2.3 Effect of Learning Depth

While deep learning is powerful in extracting features for better learning results, it is critical to decide how deep the learning network should be. We compared the performance of learning frameworks with 3 and 5 hidden layers (500, 400, 300, 200, 200 units in each layer), and the results are shown in Fig. 9a, 9d and 9e, 9h.

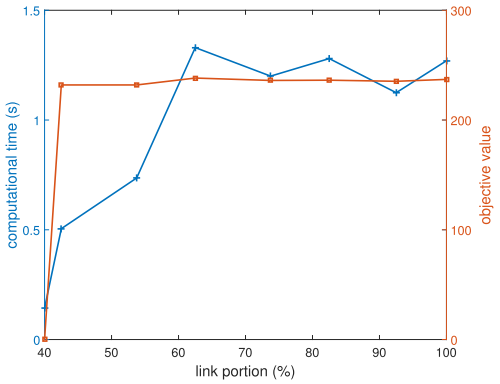
The counterintuitive result is that deeper learning framework may not necessarily lead to better performance. One possible reason is that the size of training set is limited in our case. With number of layers increased by a factor of 2, the number of parameters in the learning framework increased by 40,000 times (with 200 units per layer). Since the size of training set is unchanged, more parameters only lead to worse performance due to possible over-fitting. Generally, the proper number of layers and other parameters related to the learning framework are dependent on the training set, therefore should be adjusted accordingly.

5.2.4 Effect of Network Structure

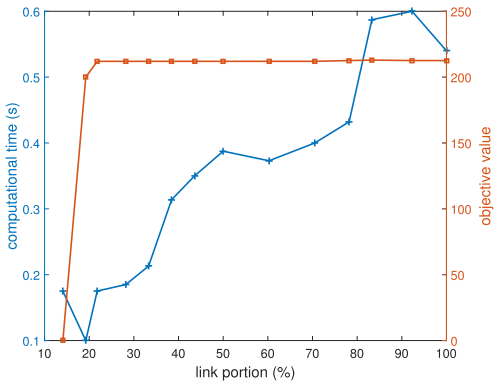
Further, by comparing the results of grid and random topology in Fig. 9, we may observe that the results of random topology can achieve nearly 100 percent detection rate with less links remaining. In other words, the learning performs better under the random topology. The reason is three-fold.

First, due to the symmetric geographic locations of grid topology, there can be multiple paths with equal performance for each flow. Which is to say, the optimal set of links (the links used in the optimal solution) is not unique. Since the data sample contains only one optimal solution, there could be alternative links with the same functionality but not present in the solution. In this case, it is likely that the learning results select the alternative links instead of the ones in the sample output, which is considered as mis-detection.

Second, in grid topology, all the links are identical in link quality. While in the random topology, some links have better transmission quality than the others and multiple flows tend to share these links for transmission. This results in different number of links used in the optimal solution for the two topologies. On average, 13.6 links are used for random topology, while 28.9 links are used for grid topology. As a



(a) Sample performance in grid topology.



(b) Sample performance in random topology.

Fig. 10. Computation time and achieved objective value under different portions of remaining links.

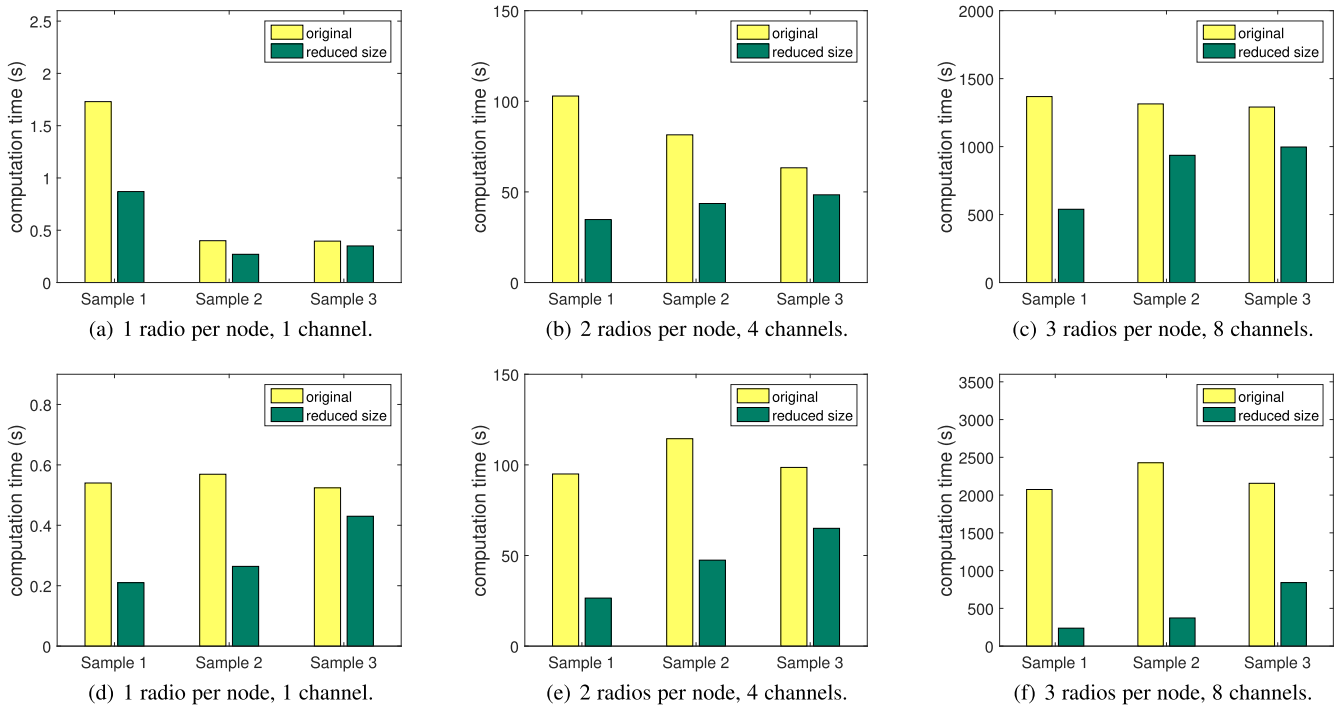
result, in order to achieve similar detection rates, more links will be kept for the grid topology since more detections are required.

Finally, as mentioned previously, whether a link will be used or not is determined by both the geographic location and link quality. In the grid topology, since there is no difference in link quality, the information that can be learned by the learning framework is limited to the locations. While in the random topology, difference in link quality can also be obtained by the learning framework. Further, there exist some "popular" links that are more frequently used. Since the training is batch based, detection of a popular link will reduce the cost function more than other links, which can be seen from Eq. 13. This means it is easier for the learning framework to detect popular links, which leads to better detection in the random topology. This observation also demonstrated that the learning framework is able to extract structural information from the network, which can be used for link evaluation.

5.3 Problem Scale Reduction

By applying the threshold to the learned link values, links with lower values than threshold will be eliminated from the formulation to reduce the optimization problem scale, as described in Algorithm 3. As mentioned previously, by adjusting the value of the threshold, different level of computation reduction can be achieved. Depending on whether the critical links are remained in the re-formulated problem, the performance in terms of the objective value (in our case, energy efficiency) of solving the reduced size problem may also be affected. The computation time and objective value under different portion of links remaining in formulation are shown in Fig. 10.

Fig. 10 demonstrated that a portion of links can be excluded from the formulation while the reduced-sized problem can yield the same objective value as that of the original problem. When too many links, including the critical links, are removed from the formulation, the performance in



(a) 1 radio per node, 1 channel. (b) 2 radios per node, 4 channels. (c) 3 radios per node, 8 channels.

(d) 1 radio per node, 1 channel. (e) 2 radios per node, 4 channels. (f) 3 radios per node, 8 channels.

Fig. 11. Computation time comparisons under different network configurations. In grid topology, Sample 1 has 48.75 percent remaining links, Sample 2 is 71.25 percent, Sample 3 is 90 percent. In random topology, Sample 1 has 33.33 percent remaining links, Sample 2 is 47.44 percent, Sample 3 is 65.38 percent. In all the cases, the objective value achieved by the reduced size problem is the same as that of the original problem.

objective value will drop or even there is no feasible solution (0 objective value in the figures). This can be avoided with a proper selection of threshold such that the computation time can be reduced without affecting the optimal solution.

The computation reduction will be more significant in larger scale optimization problems. As discussed in Section 3.3, the problem scale will be dramatically increased in a multi-dimensional resource space, such as the exploitation of multiple radios and multiple channels. The computation time comparisons under different network configurations are shown in Fig. 11, where 3 sample cases of each topology are presented with different levels of problem size reductions (resulted from different percentage of remaining links).

It can be observed that the computation reduction is effective in all the cases with up to 50 percent reduction in some cases. Fig. 11 also indicates that the cost reduction will be more significant as the scale of original problem grows. With multi-dimensional modeling technique, the optimization problem will scale more largely with the number of links, which means reducing number of links in formulation is more effective in improving computational efficiency. The above results suggest that the learning based algorithm is promising in improving the efficiency of solving large scale optimization problems.

6 Conclusion

In this paper, we have proposed a deep learning based algorithm in identifying critical links thus reducing optimization problem scale for wireless networks. Based on the observation that omitting unscheduled links can reduce problem size without affecting the solution, we have designed a deep learning framework to predict link usage by training with off-line pre-computed samples. With the link evaluation, we have proposed a learning based algorithm to exclude unused links and re-formulate the optimization problem into smaller scale. In addition, we have presented numerical results, which demonstrated that the proposed learning based method is able to detect useful links before solving the optimization problem. With the learned information, optimization problems can be solved with greatly reduced computation cost while preserving the optimality of solution, which shows the effectiveness of the proposed approach in improving efficiency of solving large scale optimization problems in wireless networks.

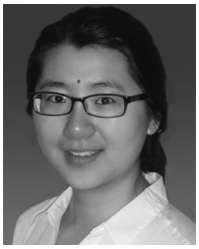
Acknowledgments

This work is supported in part by NSF grant ECCS-1610874, National Natural Science Foundation of China (NSFC) grant 61628107, and NSFC grant 61573103.

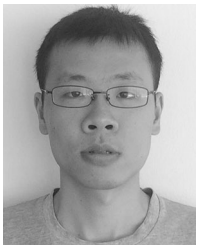
References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] X.-W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2, pp. 514–525, May 2014.
[3] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *Proc. 8th IEEE Int. Conf. Commun. Softw. Netw.*, 2016, pp. 581–585.
[4] W. Zhang, K. Liu, W. Zhang, Y. Zhang, and J. Gu, "Wi-fi positioning based on deep learning," in *Proc. IEEE Int. Conf. Inf. Autom.*, 2014, pp. 1176–1179.
[5] N. T. Nguyen, R. Zheng, and Z. Han, "On identifying primary user emulation attacks in cognitive radio systems using nonparametric bayesian classification," *IEEE Trans. Signal Process.*, vol. 60, no. 3, pp. 1432–1445, Mar. 2012.
[6] L. Liu, X. Cao, W. Shen, Y. Cheng, and L. Cai, "Dafee: A decomposed approach for energy efficient networking in multi-radio multi-channel wireless networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[7] K. K. Mensah, R. Chai, D. Bilibashi, and F. Gao, "Energy efficiency based joint cell selection and power allocation scheme for hetnets," *Digit. Commun. Netw.*, vol. 2, no. 4, pp. 184–190, 2016.
[8] L. Liu, X. Cao, Y. Cheng, L. Du, W. Song, and Y. Wang, "Energy-efficient capacity optimization in wireless networks," in *Proc. IEEE INFOCOM*, 2014, pp. 1384–1392.
[9] Y. Cheng, X. Cao, X. S. Shen, D. M. Shila, and H. Li, "A systematic study of the delayed column generation method for optimizing wireless networks," in *Proc. ACM MobiHoc*, 2014, pp. 23–32.
[10] P. Semasinghe, E. Hossain, and K. Zhu, "An evolutionary game for distributed resource allocation in self-organizing small cells," *IEEE Trans. Mobile Comput.*, vol. 14, no. 2, pp. 274–287, Feb. 2015.
[11] N. Morozs, T. Clarke, and D. Grace, "Distributed heuristically accelerated q-learning for robust cognitive spectrum management in lte cellular systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 4, pp. 817–825, Apr. 2016.
[12] H. Zhang, C. Jiang, N. C. Beaulieu, X. Chu, X. Wang, and T. Q. Quek, "Resource allocation for cognitive small cell networks: A cooperative bargaining game theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 14, no. 6, pp. 3481–3493, Jun. 2015.
[13] H. Li, Y. Cheng, C. Zhou, and W. Zhuang, "Minimizing end-to-end delay: A novel routing metric for multi-radio wireless mesh networks," in *Proc. IEEE INFOCOM*, 2009, pp. 46–54.
[14] H. Li, Y. Cheng, C. Zhou, and P. Wan, "Multi-dimensional conflict graph based computing for optimal capacity in MR-MC wireless networks," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, 2010, pp. 774–783.
[15] L. Liu, Y. Cheng, L. Cai, S. Zhou, and Z. Niu, "Deep learning based optimization in wireless network," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[16] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *J. Chemical Inf. Model.*, vol. 55, no. 2, pp. 263–274, 2015.
[17] M. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissue-regulated splicing code," *Bioinf.*, vol. 30, no. 12, pp. i121–i129, 2014.
[18] S. P. Singh and D. P. Bertsekas, "Reinforcement learning for dynamic channel allocation in cellular telephone systems," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 974–980.
[19] D. A. Tran and T. Nguyen, "Localization in wireless sensor networks based on support vector machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 7, pp. 981–994, Jul. 2008.
[20] Y. Bao, H. Wu, and X. Liu, "From prediction to action: Improving user experience with data-driven resource allocation," *IEEE J. Select. Areas Commun.*, vol. 35, no. 5, pp. 1062–1075, May 2017.
[21] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[22] L. Georgiadis, M. J. Neely, L. Tassiulas, et al., "Resource allocation and cross-layer control in wireless networks," *Found. Trends® Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
[23] Z. Han and K. R. Liu, *Resource Allocation for Wireless Networks: Basics, Techniques, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[24] J. Long, M. Dong, K. Ota, and A. Liu, "A green TDMA scheduling algorithm for prolonging lifetime in wireless sensor networks," *IEEE Syst. J.*, vol. 11, no. 2, pp. 868–877, Jun. 2017.
[25] M. Dong, K. Ota, A. Liu, and M. Guo, "Joint optimization of lifetime and transport delay under reliability constraint wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 225–236, Jan. 2016.
[26] D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Belmont, MA, USA: Athena Scientific, 1998.
[27] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[28] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends® Mach. Learn*, vol. 3, no. 1, pp. 1–122, 2011.
[29] Z. Chang, Z. Wang, X. Guo, Z. Han, and T. Ristaniemi, "Energy efficient and distributed resource allocation for wireless powered ofdma multi-cell networks," in *Proc. 15th Int. Symp. Model. Optimization Mobile Ad Hoc Wireless Netw.*, 2017, pp. 1–6.
[30] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.
[31] B. Yin, W. Shen, Y. Cheng, L. X. Cai, and Q. Li, "Distributed resource sharing in fog-assisted big data streaming," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.
[32] G. Liu, F. R. Yu, H. Ji, and V. C. Leung, "Distributed resource allocation in virtualized full-duplex relaying networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 8444–8460, Oct. 2016.
[33] Z. Han, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[34] L. Liu, Y. Cheng, X. Cao, S. Zhou, and Z. Niu, "Joint optimization of scheduling and power control in wireless network: Multidimensional modeling and decomposition," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1585–1600, Jul. 2019.
[35] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.



Lu Liu (S'13-M'18) received the PhD degree in computer engineering from Illinois Institute of Technology, Chicago, Illinois, in 2017. Her current research interest includes energy efficient networking and communication, resource allocation and protocol design of wireless networks, and machine learning based network optimization. She is a member of the IEEE.



Bo Yin (S'14) received the BE degree in electronic information engineering and the ME degree in electronic science and technology from Beihang University, in 2010 and 2013, respectively. Currently, he is working toward the PhD degree in the Department of Electrical and Computer Engineering, Illinois Institute of Technology. His research interests include edge computing and network security. He is a student member of the IEEE.

Shuai Zhang (S'15) received the BEng and MS degrees from Zhejiang University and the University of California, Los Angeles, in 2013 and 2015, respectively. He is working toward the PhD degree at Illinois Institute of Technology. His research interests include wireless communication and distributed computing. He is a student member of IEEE.

Xianghui Cao (S'08-M'11-SM'16) received the BS and PhD degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively. Currently he is an associate professor with the School of Automation, Southeast University, Nanjing, China. His research interests include cyber-physical systems, wireless network performance analysis, wireless networked control and network security. He is a senior member of the IEEE.

Yu Cheng (SM'09) received the BE and ME degrees in electronic engineering from Tsinghua University, in 1995 and 1998, respectively, and the PhD degree in electrical and computer engineering from the University of Waterloo, Canada, in 2003. He is now a full professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology. His research interests include wireless network performance analysis, network security, big data, and cloud computing.