A Machine Learning-Based Algorithm for Joint Scheduling and Power Control in Wireless Networks

Xianghui Cao, *Senior Member, IEEE*, Rui Ma, Lu Liu, *Member, IEEE*, Hongbao Shi, Yu Cheng, *Senior Member, IEEE*, and Changyin Sun

***Abstract*—Wireless network resource allocation is an important issue for designing Internet of Things systems. In this paper, we consider the problem of wireless network capacity optimization that involves issues such as flow allocation, link scheduling, and power control. We show that it can be decomposed into a linear program and a nonlinear weighted sum-rate maximization problem for power allocation. Unlike most traditional methods that iteratively search the optimal solutions of the nonlinear subproblem, we propose to directly compute approximated solutions based on machine learning techniques. Specifically, the learning systems consist of both support vector machines (SVMs) and deep belief networks (DBNs) that are trained based on offline computed optimal solutions. In the running phase, the SVMs perform classification for each link to decide whether to use maximal transmit power or be turned off. At the same time, the DBNs compute an approximation of the optimal power allocation. The two results are combined to obtain an approximated solution of the nonlinear program. Simulation results demonstrate the effectiveness of the proposed machine learning-based algorithm.**

***Index Terms*—Deep belief network (DBN), deep learning, joint optimization, power control, scheduling, support vector machine (SVM), wireless network.**

I. Introduction

MACHINE learning techniques have been widely applied in various areas such as data mining, computer vision, ranking or recommendation systems, and pattern or abnormality detection [1]–[5]. Recently, machine learning has also found its applications in wireless networks to address complex computation issues raised by the large network size or high big data volumes [6]–[11]. Most of such applications, however, focus on utilizing machine learning techniques to deal with the data sets themselves as those done in traditional data analysis applications. It is not fully explored what role machine learning can play on those fundamental research issues in wireless networks such as scheduling, resource allocation, and optimization. In this paper, we make an effort to study how to exploit machine learning for solving complex optimal resource allocation issues in wireless networks.

In the literature, machine learning has been exploited from different aspects to address networking issues, with the objectives of enhancing performance, developing new protocols, or reducing computation complexity. For example, deep neural networks have been established in [8] to solve routing problems in dynamic traffic environments. Machine learning techniques also have been applied in MAC protocol design [12], quality-of-service-aware resource allocation [13], network traffic classification [14], flow prediction [15], and mobility prediction [16]. Despite the various applications of machine learning in networking, only a few works lay focus on network optimizations in the sense of scheduling and resource allocation. The work in [17] and [18] implements machine learning in wireless network optimization, by performing learning-based link evaluations to exclude unused links from problem formulation, so that the computational complexity can be reduced without affecting optimal solutions.

In this paper, we consider the classic yet challenging research issue: how to maximize network capacity through link scheduling and power control, under a physical interference model [19], [20]. In order to obtain a joint optimization solution, existing approaches usually introduce high-complexity algorithms or long convergence time due to the NP hardness nature, where many coupling network resources incur high dimensional optimization variables and constraints. To speed up the convergence, people resort to relaxing the original problem, developing polynomial approximation algorithms, or distributed algorithms based on game theory, e.g., [19] and [21]–[23]. However, the solutions in these works either sacrifice optimality for fast convergence, or suffer from long convergence time in order to reach an optimum. In this paper, we take an innovative perspective that the capability of machine learning can be exploited to address such challenging network optimization problem, to achieve a good balance of performance and complexity.

Machine learning can be viewed as a sophisticated tool to extract information and knowledge from the training process

Manuscript received January 2, 2018; revised May 27, 2018; accepted June 15, 2018. Date of publication July 6, 2018; date of current version January 16, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61573103, Grant 61628107, and Grant 61520106009, in part by the U.S. National Science Foundation (NSF) under Grant ECCS-1610874, in part by the State Key Laboratory of Synthetical Automation for Process Industries, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Changyin Sun.)*

X. Cao, R. Ma, H. Shi, and C. Sun are with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: xhcao@seu.edu.cn; mrui0316@seu.edu.cn; hbshi@seu.edu.cn; cysun@seu.edu.cn).

L. Liu and Y. Cheng are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: lliu41@hawk.iit.edu; cheng@iit.edu).

Digital Object Identifier 10.1109/JIOT.2018.2853661

based on empirical data. Generally, the results of network optimization (i.e., the maximum network capacity and associated optimal resource allocation) are determined by the input of the problem, such as network topology, flow demands, and resource constraints. With such a perspective, the process of solving the optimization problem can be viewed as investigating the relationship between network input and output, which is usually so complicated and impossible to be characterized with an explicit function expression. Nevertheless, such a task may be fulfilled with machine learning techniques for their capabilities in extracting high-level abstractions from input data and regression of complicated function. The training data can be obtained from computation experiences, which are the formulation and solutions of historical optimization problems. Based on the training data, a learning framework can be built and trained to extract the desired latent information. Then given a new input (yielding a new optimization problem), the learning framework can respond with an output with utilizations of the learned knowledge such as an estimated function mapping from input to output.

In this paper, we formulate the capacity maximization problem as a joint nonlinear program of flow allocation, link scheduling, and power control. The problem is then decomposed into a scheduling and flow control subproblem (master problem) which is a linear program and a power control subproblem (slave problem) which is nonlinear. The two subproblems can be solved recursively to achieve the optimum of the original capacity optimization problem. We show that the slave subproblem is equivalent to the weighted sum-rate maximization problem, which reflexes the main complexity of solving the original problem. Unlike traditional methods to iteratively search the optimal solutions of the slave subproblem, we propose to apply machine learning techniques to directly approximate the optimal subproblem solution given the input from the solution of the master problem. Specifically, we establish a number of support vector machines (SVMs) and deep belief networks (DBNs) to learn the solution structure of the power control subproblem. The proposed method consists of three phases, namely data preparation, training, and running phases. The learning systems are trained based on offline optimal solution samples obtained by off-the-shelf optimization tools. After training and in the running phase, given an input from the master problem, the SVMs are first applied to classify the extreme cases, i.e., whether a link should be assigned the maximum or 0 power. Then, the DBNs are used to obtain fine-grained power allocations. The two results are combined in a probabilistic manner to output the solution to the master problem. The proposed method is evaluated and compared with other methods through simulations.

Our main contributions can summarized as follows.

1) We explore machine learning techniques to address complex resource allocation problems in wireless networks.
2) We decompose the network capacity optimization problem and propose a machine learning-based method integrated with SVMs and DBNs to solve the nonlinear part of the optimization problem.
3) Simulation results demonstrate that the proposed learning-based method achieves close performance to an optimal iterative algorithm but at a much lower cost.

The remainder of this paper is organized as follows. Section II reviews more related work. Section III presents the network models and formulates the capacity optimization problem. Section IV decomposes the problem. The learning-based algorithm is presented in Section V. Section VI presents the simulation results and Section VII concludes this paper.

II. Related Work

In this paper, we show that the main difficulty in optimally solving the decomposed capacity optimization problem lies in the complexity of the nonlinear subproblem, which is known as the nonconvex weighted sum-rate maximization problem. In the literature, this problem and its closely related variant—the max–min rate problem—have been studied, e.g., [19], [24], and [25]. In [21], the sum-rate maximization problem is transformed into a multiplicative linear fractional programming, which can be solved by an iterative algorithm. Tan *et al.* [19] applied nonnegative matrix inequalities to transform the original problem with power as the variables to one that directly optimizes the signal-to-noise-plus-interference ratios (SINRs). They show that the problem can further be transformed into a convex optimization and the optimal solutions are achieved at the boundary of a convex set. Instead of directly solving such a problem, two relaxation techniques are used and two iterative algorithms are proposed to achieve near optimal solutions. Similarly, a geometrically fast convergence algorithm with approximately optimal solutions is proposed in [22], and a suboptimal scheduling policy under congestion control is obtained to solve the sum-rate maximization problem. However, such iterative algorithms require a long convergence time until an optimal (or a suboptimal) solution can be achieved. Both exact computational tools with column generation and approximation alternatives are proposed in [24], for maximizing minimum throughput in wireless mesh networks. These algorithms can achieve near optimal solutions with less time, but at the cost of loss in optimality.

The main idea of this paper is to apply machine learning techniques to approximate the optimal solutions of the nonlinear subproblem of power control. In the literature, there are some related works on learning-based approaches for various issues of networking. Mao *et al.* [8] proposed a deep learning-based routing algorithm for high-speed core networks where they explore learning capabilities of smart routers to directly compute routing decisions. In this sense, compared to traditional rule-based routing policies, the learning-based method can reduce signaling overhead and the routing decisions can better respond to network traffic dynamics. The proposed learning system in [8] consists of multiple restricted Boltzmann machines (RBMs) with the input of the bottom RBM as the traffic pattern and the output of the top RBM as the routing decision vector. Machine learning-based MAC protocols have been investigated in [12] for energy saving, where the energy effectiveness is estimated with Bayesian method under conjugate prior information. For quality of experience

TABLE I
ABBREVIATIONS AND NOTATIONS

Term	Meaning
SVM	Support vector machine
DBN	Deep belief network
SINR	Signal-to-noise-plus-interference ratio
RBM	Restricted Boltzmann machine
MLA	The proposed machine learning based algorithm
GA	Genetic algorithm
$\mathcal{N}$	Set of all nodes in the network
$\mathcal{L}$	Set of all physical links
$\mathcal{F}$	Set of all commodity flows
$\text{Tx}(\ell), \text{Rv}(\ell)$	Transmitting and receiving nodes of link ℓ, respectively
$s(f), d(f)$	Source and destination nodes of flow f, respectively
q_f	Rate demand of flow f by user
p_ℓ	Transmit power of link ℓ. $p_\ell \in [0, \bar{p}_\ell]$
$\mathbf{p}$	The power allocation vector. $\mathbf{p} = (p_1, p_2, \ldots, p_{\lvert\mathcal{L}\rvert})'$
α_i	The portion of time allocated to a power allocation $\mathbf{p}_i$
$\gamma_\ell, \lambda_\ell$	The SINR and achieved capacity of link ℓ, respectively
$r_{f,\ell}$	The rate allocated to link ℓ for flow f
θ	The network capacity under fairness consideration

provisioning in mobile networks, a dynamic resource allocation scheme is proposed in which various machine learning techniques (e.g., bagging tree, boosted tree, K-nearest neighbor, SVM, and Kohonen networks) are applied to predict the network performance based on measurement data from users such as SINR, throughput and delay [13]. The authors show that bagging tree outperforms others in their case. Machine learning is also adopted in areas such as network traffic classification [14], flow prediction [15], and mobility prediction [16], to fulfill traffic analysis and control. A model-free reinforcement learning algorithm is presented in [26] for scheduling and resource allocation schemes in heterogeneous networks with hybrid energy to maximize energy efficiency in the network.

III. Problem Formulation

We consider a static wireless network described by a graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ is the set of all network nodes while $\mathcal{L}$ represents the set of all (directed) physical links between the nodes. For any link $\ell \in \mathcal{L}$, denote $\text{Tx}(\ell)$ and $\text{Rv}(\ell)$ as the transmitting and receiving nodes of ℓ, respectively. Suppose that there are a set $\mathcal{F}$ of commodity flows traversing in the network with each flow specified by a pair of source and destination nodes. Each flow has a rate demand of q_f by users. Our objective is to maximize the network capacity to best satisfy user demands while keeping a certain level of fairness. This requires jointly solving the coupled issues of flow allocation, link transmission time scheduling and transmit power control. The routing information can be inferred from the flow allocation and link scheduling results, to be explained later. Hereafter, we assume that each flow has at least one path from its source to its destination; otherwise, the flow demand cannot be achieved. Table I summarizes the main abbreviations and notations used throughout this paper.

Throughout this paper, we adopt the following notations: we use lower-case letters in boldface to represent vectors (column vectors by default) and capital letters to denote matrices or constant numbers. All numbers, vectors, and matrices in this paper take real values. For a vector $\mathbf{v}$, $[\mathbf{v}]_i$ is its ith element; similarly, $[M]_{i,j}$ is the (i, j)th entry of matrix M. For a matrix M, M' is its transpose while $\rho(M)$ is its spectral radius if M is a square matrix. I is the identify matrix whose dimension is clear in the context. For a vector $\mathbf{v}$ of dimension m, $\text{diag}(\mathbf{v})$ is a $m \times m$ diagonal matrix with ith diagonal entry equals to $[\mathbf{v}]_i$. For two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ of the same dimension, $\mathbf{v}_1 \succcurlyeq \mathbf{v}_2$ represents that $\forall i, [\mathbf{v}_1]_i \geq [\mathbf{v}_2]_i$. The operator $\preccurlyeq$ is similarly defined. $|\cdot|$ denotes the number of elements of a set.

A. System Models

1) Transmit Power Control: We adopt the following physical interference model to characterize the relationships among links. For any link $\ell \in \mathcal{L}$, let $p_\ell \in [0, \bar{p}_\ell]$ be the transmit power of $\text{Tx}(\ell)$. Denote $\mathbf{p} = (p_1, p_2, \ldots, p_{|\mathcal{L}|})'$ as a power allocation vector for all the links. Then, $\forall \ell \in \mathcal{L}$, its SINR is given by

$$\gamma_\ell(\mathbf{p}) = \frac{g_{\ell,\ell} p_\ell}{\sigma_\ell + \sum_{l \in \mathcal{L} \setminus \ell} g_{l,\ell} p_l} \tag{1}$$

where σ_ℓ is the average noise power on ℓ. $\forall \ell, l \in \mathcal{L}$, $g_{l,\ell}$ means the channel gain from the transmitter of l to the receiver of ℓ, which depends on the path loss, shadowing, and fading factors. Particularly, $g_{\ell,\ell}$ is the gain of link ℓ. A power allocation should avoid the impractical situations that two conflicting links are simultaneously scheduled (or equivalently are allocated positive power) if: 1) they share the same transmitter and 2) the receiver of one link is the transmitter of the other, due to the half-duplex constraint. Hence, at any time, if a node is scheduled to transmit data, at most one of its outbound links is scheduled. To characterize such conflicts, we define a matrix C with each entry as

$$c_{l,\ell} = \begin{cases} 1, & \text{if } \{\text{Tx}(l), \text{Rv}(l)\} \text{ and } \{\text{Tx}(\ell), \text{Rv}(\ell)\} \text{ share a} \\ & \text{common node, except that } \text{Rv}(l) = \text{Rv}(\ell) \\ 0, & \text{otherwise} \end{cases}$$

and the conflicting constraint of two links can be captured by

$$c_{l,\ell} p_l p_\ell = 0 \qquad \forall l; \ell \in \mathcal{L}. \tag{2}$$

In the following, we say a power allocation $\mathbf{p}$ is feasible if the above equation holds.

Given a feasible power allocation $\mathbf{p}$, the capacity (maximum achievable rate) of a link ℓ can be calculated as follows:

$$\lambda_\ell(\mathbf{p}) = \begin{cases} B \log(1 + \gamma_\ell(\mathbf{p})), & \text{if } \gamma_\ell(\mathbf{p}) \geq \underline{\gamma}_\ell \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where B is the channel bandwidth. $\underline{\gamma}_\ell$ denotes the SINR threshold of link ℓ below which the received signal cannot be successfully decoded by the receiver.

2) Link Schedule: Since $\mathbf{p}$ is continuous, there are infinitely many possible power allocations each of which can allow multiple links, if not conflicting, to transmit data concurrently. For the problem tractability and that practically we cannot schedule uncountably many power allocations, we consider a finite number of allocations for ease of presenting the mathematic problem formulation. Let $P = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K]$ be the matrix of a sufficiently large number of feasible power allocations. Then, the scheduling subproblem is to allocate the transmission time for each $\mathbf{p}_i$. Consider a finite scheduling time horizon and define α_i as the portion of total time allocated to $\mathbf{p}_i$. Denote $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]$. We must have

$$\sum_{i=1}^{K} \alpha_i = 1. \tag{4}$$

3) Network Flow Control: For each flow f, define a $|\mathcal{N}| \times |\mathcal{L}|$ incidence matrix H_f where $\forall i \in \mathcal{N}$ and $\forall \ell \in \mathcal{L}$

$$[H_f]_{i,\ell} = \begin{cases} 1, & \text{if } i = \text{Tx}(\ell) \\ -1, & \text{if } i = \text{Rv}(\ell) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Let $\mathbf{r}_f = [r_{f,1}, \ldots, r_{f,|\mathcal{L}|}]'$ be the flow allocation vector for flow f. Then, if the demand of f is exactly satisfied, we have

$$H_f \mathbf{r}_f = \mathbf{q}_f \tag{6}$$

where $\forall i \in \mathcal{N}$, the ith element of vector $\mathbf{q}_f$ is

$$[\mathbf{q}_f]_i = \begin{cases} q_f, & \text{if } i = s_f \\ -q_f, & \text{if } i = d_f \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Equation (6) is called the flow balance constraint, which ensures that the import and export rates of a flow are balanced at each nonsource and nondestination node. In this paper, considering the fairness among all the flows, we introduce the scaling factor θ for network capacity optimization in the sense that the final achieved rate of each flow is θ times of its demand, i.e., $\forall f \in \mathcal{F}$

$$H_f \mathbf{r}_f = \theta \mathbf{q}_f. \tag{8}$$

Moreover, the flow allocated to each link is upper-bounded by its capacity after scheduling, i.e., $\forall \ell \in \mathcal{L}$

$$\sum_{f \in \mathcal{F}} r_{f,\ell} \le \sum_{i=1}^{K} \alpha_i \lambda_\ell(\mathbf{p}_i). \tag{9}$$

According to [27], we can always choose $\boldsymbol{\alpha}$ such that the left-hand side of (9) exactly meets the right, which will then benefit the network capacity. Therefore, in the following, (9) is treated as an equality.

B. Problem Formulation

With the above models, the joint transmit power control, scheduling and flow allocation problem for capacity optimization under fairness consideration is formulated as follows.

Problem 1 (Original Problem):

$$\max_{\boldsymbol{\alpha}, \mathbf{r}, P} \quad \theta \tag{10}$$

$$\text{s.t.} \quad (1)\text{–}(4),\ (8),\ (9) \tag{11}$$

$$0 \le p_\ell \le \bar{p}_\ell \qquad \forall \ell \in \mathcal{L} \tag{12}$$

$$\alpha_i \ge 0 \qquad \forall i = 1, 2, \ldots \tag{13}$$

$$r_{f,\ell} \ge 0 \qquad \forall \ell \in \mathcal{L}; \forall f \in \mathcal{F} \tag{14}$$

$$\left(\gamma_\ell(\mathbf{p}_i) - \underline{\gamma}_\ell\right) p_\ell \ge 0 \qquad \forall \ell \in \mathcal{L}; \forall i = 1, 2, \ldots \tag{15}$$

where the last inequality ensures that either $p_\ell = 0$ or both $p_\ell > 0$ and $\gamma_\ell \ge \underline{\gamma}_\ell$ for each ℓ. Notice that the solutions of $\mathbf{r}$ and $\boldsymbol{\alpha}$ together tell that when and at what rate a link transmits which flow's data. At the same time, the flow balance constraint in (8) ensures that a flow from its source will be delivered to its destination. Therefore, for each flow, the scheduled links will connect the source and destination nodes, which indicate the routes the flow traverses.

The above problem is a nonlinear program due to the nonlinearity in (1) and (3). Furthermore, its difficulty also lies in the impossibility of enumerating all the power allocations $\{\mathbf{p}\}$. In the following, we resort to a problem decomposition technique to develop an effective algorithm.

IV. Problem Decomposition

The decomposition is based on column generation technique [27]–[29]. Instead of enumerating all the power allocations, we gradually insert new power allocations to existing set of allocations step-by-step. Let P_k be the matrix of all feasible power allocations obtained by step $k-1$ and $\mathbf{p}_k$ be the new feasible allocation found in step k to be added to P_k. Let n_k be the number of allocations (columns) in P_k. Then

$$P_{k+1} = L\big[P_k, \mathbf{p}_k\big] \quad \text{and} \quad n_{k+1} = n_k + 1.$$

We start with a small set of feasible allocations, say $P_0 = [\mathbf{p}_1^0, \ldots, \mathbf{p}_{n_0}^0]$. At any step k with a given P_k, Problem 1 reduces to a joint scheduling and flow allocation problem which can be represented in a matrix form as follows.

Problem 2 (Master Subproblem): The fixed-power joint scheduling and flow allocation problem at step k is

$$\max_{\mathbf{x}_k} \quad \mathbf{u}_k' \mathbf{x}_k \tag{16}$$

$$\text{s.t.} \quad A_k \mathbf{x}_k = \mathbf{b} \tag{17}$$

$$\mathbf{x}_k \succcurlyeq \mathbf{0} \tag{18}$$

where

$$\mathbf{x}_k = \left[\theta, \tilde{\mathbf{r}}_k', \boldsymbol{\alpha}_k'\right]' \tag{19}$$

$$\mathbf{u}_k = \left[1, \mathbf{0}'_{|\mathcal{L}||\mathcal{F}|+n_k}\right]' \tag{20}$$

$$\mathbf{b} = \left[\mathbf{0}'_{|\mathcal{N}||\mathcal{F}|+|\mathcal{L}|}, 1\right]' \tag{21}$$

$$A_k = \left[\begin{array}{c|cccc|c} -\mathbf{q}_1 & H_1 & & & & \\ \vdots & & \ddots & & & \\ -\mathbf{q}_{|\mathcal{F}|} & & & & H_{|\mathcal{F}|} & \\ \hline & -I_1 & & \ldots & -I_{|\mathcal{F}|} & \boldsymbol{\lambda}(P_k) \\ & & & & & \mathbf{1}'_{n_k} \end{array}\right]. \tag{22}$$

In the above, $\tilde{\mathbf{r}} = [\mathbf{r}_1', \ldots, \mathbf{r}_{|\mathcal{F}|}']'$. I_f is the identity matrix of dimension $|\mathcal{L}| \times |\mathcal{L}|$. $\mathbf{0}_n$ is an n-sized vector with all elements equal to 0. $\mathbf{1}_n$ is similarly defined. $\boldsymbol{\lambda}(P_k) = [\boldsymbol{\lambda}(\mathbf{p}_1), \ldots, \boldsymbol{\lambda}(\mathbf{p}_{n_k})]$ and $\boldsymbol{\lambda}(\mathbf{p}_i) = [\lambda_1(\mathbf{p}_i), \ldots, \lambda_{|\mathcal{L}|}(\mathbf{p}_i)]'$.

Clearly, Problem 2 is a linear program that can be efficiently solved. Starting from the initial allocation set P_0, by (22), adding a new $\mathbf{p}_k$ to P_k is equivalent to adding a new column $\mathbf{a}_k = [\mathbf{0}'_{|\mathcal{N}||\mathcal{F}|}, \boldsymbol{\lambda}'(\mathbf{p}_k), 1]'$ to form $A_{k+1} = [A_k, \mathbf{a}_k]$. Based on the column generation method [28], $\mathbf{a}_k$ is chosen as the one that maximizes the reduced cost as follows:

$$-\tilde{\mathbf{w}}_k' \mathbf{a}_k = -\tilde{w}_{|\mathcal{N}||\mathcal{F}|+|\mathcal{L}|+1} - \sum_{\ell=1}^{|\mathcal{L}|} \tilde{w}_{k,|\mathcal{N}||\mathcal{F}|+\ell} \lambda_\ell(\mathbf{p}_k) \tag{23}$$

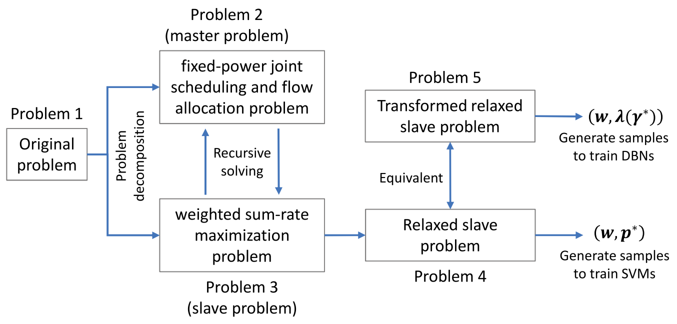

Problem 2
(master problem)
fixed-power joint scheduling and flow allocation problem
Problem 1
Original problem
Problem decomposition
Recursive solving
weighted sum-rate maximization problem
Problem 3
(slave problem)
Problem 5
Transformed relaxed slave problem
Equivalent
Relaxed slave problem
Problem 4
$(\boldsymbol{w}, \boldsymbol{\lambda}(\boldsymbol{\gamma}^*))$
Generate samples to train DBNs
$(\boldsymbol{w}, \boldsymbol{p}^*)$
Generate samples to train SVMs

Fig. 1. Relationships among the problems studied in this paper.

where $\tilde{\mathbf{w}}_k$ is the optimal dual associated with the constraint in (22). Let $\mathbf{w}_k = [-\tilde{w}_{k,|\mathcal{N}||\mathcal{F}|+1}, \ldots, -\tilde{w}_{k,|\mathcal{N}||\mathcal{F}|+|\mathcal{L}|}]'$. Then, we have the following theorem.

Theorem 1: To find a new feasible power allocation $\mathbf{p}_k$ to add to P_k is equivalent to solving a weighted sum-rate maximization problem under a conflicting constraint and a minimum SINR constraint as stated in Problem 3.

Problem 3 (Slave Subproblem):

$$\max_{\mathbf{p}} \quad \mathbf{w}_k'\boldsymbol{\lambda}(\mathbf{p}) \tag{24}$$

$$\text{s.t.} \quad \mathbf{0} \preccurlyeq \mathbf{p} \preccurlyeq \bar{\mathbf{p}} \tag{25}$$

$$\text{diag}(\mathbf{p})C\text{diag}(\mathbf{p}) = 0 \tag{26}$$

$$\text{diag}(\mathbf{p})\Big(\boldsymbol{\gamma}(\mathbf{p}) - \underline{\boldsymbol{\gamma}}\Big) \succcurlyeq \mathbf{0}. \tag{27}$$

This paper considers several problems for which the relationships are displayed in Fig. 1. The original optimization problem (i.e., Problem 1) is decomposed into a master problem (Problem 2) and a slave problem (Problem 3) which is a nonlinear. Problems 2 and 3 are solved recursively. In each iteration, we first solve Problem 2, which outputs $\mathbf{w}_k$ to form Problem 3. Then the latter is solved, which outputs a new power allocation vector $\mathbf{p}_k$ that is then used to update Problem 2. The iterations stop when no new $\mathbf{p}_k$ can be found, i.e., the achieved objective in Problem 3 is nonpositive, indicating that an optimal solution to Problem 1 is obtained.

Remark 1: Since $\lambda_\ell(\mathbf{p}) \geq 0$, the optimal power for link ℓ is 0 if $w_\ell \leq 0$. Therefore, for those $w_\ell \leq 0$, we can simply set them 0, which will not change the optimal solution of Problem 3. Meanwhile, the optimal solution also remains if the weights in $\mathbf{w}_k$ are normalized by dividing their maximum value. Thus, in the following, it is without loss of generality to assume that $w_\ell \in [0, 1]$ for all ℓ in the above problem.

The initial power allocation matrix P_0 can be arbitrarily given as long as all $\{\mathbf{p}_i^0\}$ are feasible (i.e., no conflicting links are scheduled at the same time) and cover all links (i.e., each link is scheduled in at least one of $\{\mathbf{p}_i^0\}$). Therefore, a simple method is to set $P_0 = [\bar{p}_1\mathbf{e}_1, \ldots, \bar{p}_{|\mathcal{L}|}\mathbf{e}_{|\mathcal{L}|}]$ with $n_0 = |\mathcal{L}|$, where $\mathbf{e}_i$ is a $|\mathcal{L}|$-sized vector with ith element equals to 1 and all the others equal to 0.

Problem 3 is a nonconvex nonlinear program which is still difficult to solve. In the literature, there are some iterative algorithms developed to solve this sum-rate maximization problem effectively [19], [25]. However, such algorithms usually require a long time to converge. In the following section, we propose a learning-based method to directly provide an approximated solution of Problem 3.

V. Learning-Based Algorithm

An overview of the proposed learning-based method is illustrated in Fig. 2. The main idea is to establish a set of SVMs and neural networks to approximate the optimal solutions of Problem 3 by offline learning a large set of optimal solution samples. The method consists of three phases. In the first data preparation phase, we apply some off-the-shelf nonlinear programming algorithm to solve Problem 3 and obtain a sufficient number of samples. Each sample is denoted as a pair $(\mathbf{w}, \mathbf{y})$, where $\mathbf{y}$ is a function of the optimal solution given $\mathbf{w}$. In the second phase, a set of SVMs and DBNs are established and trained by the obtained samples $\{(\mathbf{w}, \mathbf{y})\}$, where the input of each SVM and DBN is $\mathbf{w}$ while the output is $\{0, 1\}$ for SVM and y_ℓ for DBN. The SVMs and DBNs after training will be used to approximate the solutions of Problem 3 online in the third phase. The whole learning-based algorithm [machine learning-based algorithm (MLA)] that iteratively solves the original problem is summarized in Algorithm 1.

A. Preliminaries

Before presenting the detailed algorithm design, we first briefly introduce the SVM and DBN. SVM is a supervised learning method normally for classification, i.e., to decide the category a new data point best belongs to [30]. Consider the linear classification case. For a set of training data that can be viewed as points in some space, given that each point is labeled by either one of two categories, two subsets of the points are formed. SVM is to find the maximum-margin hyperplane that separates the two subsets with the largest separation. In other words, training a linear SVM is to determine a hyperplane that maximizes the distances from it to the two subsets. Detailed designs are presented in Section V-C1.

DBN is an effective class of deep learning networks [31]. As shown in Fig. 3, an example of the structure of DBN consists of an input layer, a number of hidden layers, and an output layer, which is also a stack of RBMs. Each RBM is an undirected, generative energy-based model consisting of a visible layer and a hidden layer. As shown in this figure, the bottom RBM takes the input layer as its visible layer. The next RBM takes the hidden layer of the previous RBM as its visible layer and the next hidden layer as its own hidden layer. The neurons in different layers are connected with each connection associated with a weight, while those belonging to the same layer are disconnected. The training process for each DBN contains two steps. In the first step, the RBMs are trained layer-by-layer with unsupervised learning, while in the second step the whole DBN is fine-tuned with supervised learning based on the backpropagation method. Detailed DBN design and training process are presented in Section V-C2.

B. Data Preparation Phase to Obtain Training Sets

We need to obtain a sufficient number of sample optimization solutions in order to train the machine learning systems. Notice that the objective function in Problem 3

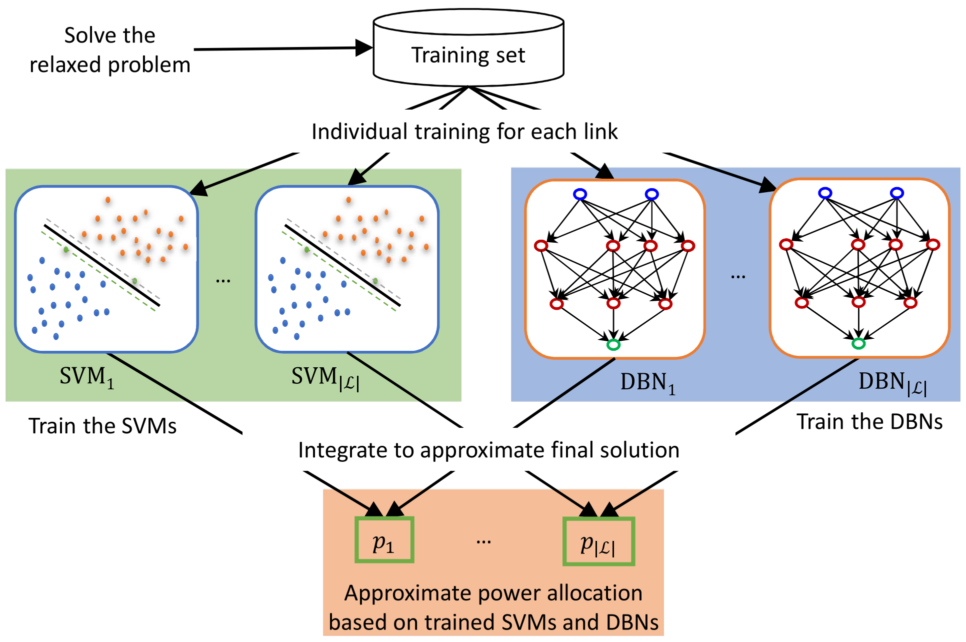

Solve the relaxed problem
Training set
Individual training for each link
SVM_1
$\text{SVM}_{|\mathcal{L}|}$
DBN_1
$\text{DBN}_{|\mathcal{L}|}$
Train the SVMs
Train the DBNs
Integrate to approximate final solution
p_1
$p_{|\mathcal{L}|}$
Approximate power allocation based on trained SVMs and DBNs

Fig. 2. Structure of the learning system based on SVMs and DBNs.

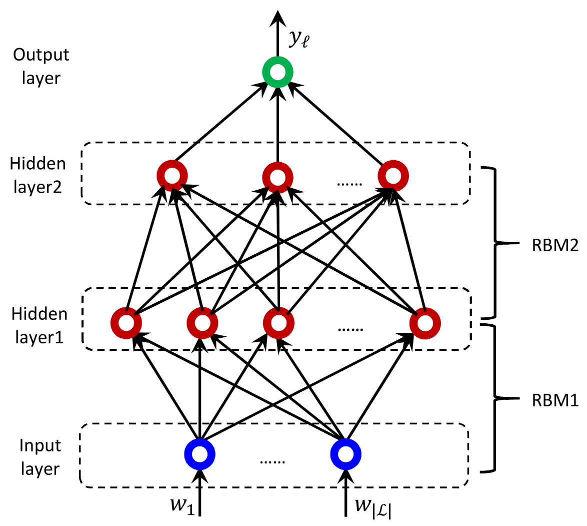


Fig. 3. Illustration of the structure of each DBN.

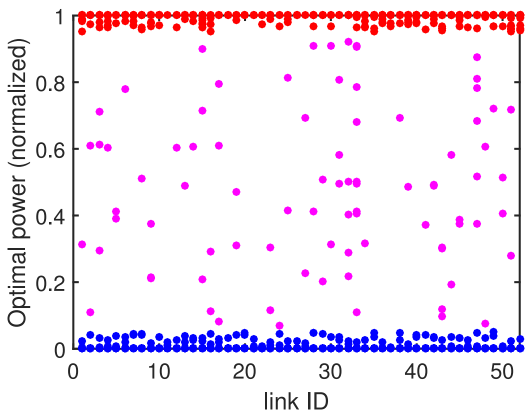


Fig. 4. Typical distribution of the optimal power solutions, where different optimal power solutions are plotted which are calculated by using the GA based on many randomly generated weights $\mathbf{w}$.

is nonlinear and contains nonsmooth points [see (3)]. Since $\underline{\gamma}_\ell$ usually is small, we relax Problem 3 by ignoring $\{\underline{\gamma}_\ell\}$ as follows.

Problem 4 (Relaxed Slave Problem):

$$\max_{\mathbf{p}} \sum_{\ell\in\mathcal{L}} w_{k,\ell} \log(1+\gamma_\ell(\mathbf{p})) \tag{28}$$

$$\text{s.t.}\quad \mathbf{0} \preccurlyeq \mathbf{p} \preccurlyeq \bar{\mathbf{p}} \tag{29}$$

$$\text{diag}(\mathbf{p})C\text{diag}(\mathbf{p}) = 0. \tag{30}$$

The optimal solutions to the above problem can be searched by some optimization algorithms such as genetic algorithms (GAs) and simulated annealing. In our simulations, the GA is adopted to generated sample solutions. As shown in Fig. 4, from a typical distribution of the optimal power solutions, we can observe that the optimal power of most links are either within a small range of 0 or close to their maximal value $\{\bar{p}_\ell\}$. This is reasonable since in the optimal solution, interference is minimized in order to improve the link rates, likely resulting in either small or large power allocations. Besides, since the threshold $\underline{\gamma}_\ell$ is ignored in Problem 4, the optimal solution may yield very small p_ℓ^*'s. In view of this, we apply SVMs to classify the power allocation for each link into three categories, i.e., close to 0, close to its maximal transmit power and away from the two extreme values, in order to provide a first-step rough approximation of the optimal power allocation. In this case, the samples of input weight and optimal solution pairs (i.e., $\{(\mathbf{w}, \mathbf{y}_{\text{SVM}})\}$ with $\mathbf{y}_{\text{SVM}} = \mathbf{p}^*$) are used to train the SVMs, where $\mathbf{p}^*$ is the optimal solution of Problem 4.

In order to refine the optimal power approximation, instead of dwelling on $\{(\mathbf{w}, \mathbf{p}^*)\}$, we directly use the link SINRs to train the neural networks. We apply deep learning here in order to take the potential advantage of its superior performance in many fields such as signal processing, computer vision, bioinformatics, and so on. DBN is a typical and one of the first effective deep learning algorithms. In this paper, we apply DBN to learn the data of link weights (input) and SINRs (output). First, we transform the above problem into a new problem that directly optimizes the SINRs. Define the following vectors and matrix:

$$\begin{aligned}\mathbf{g} &= \left[g_{1,1}, \ldots, g_{|\mathcal{L}|,|\mathcal{L}|}\right]' \\ \tilde{\boldsymbol{\sigma}} &= \left[\frac{\sigma_1}{g_{1,1}}, \ldots, \frac{\sigma_{|\mathcal{L}|}}{g_{|\mathcal{L}|,|\mathcal{L}|}}\right]' \\ G &= \left[\frac{g_{i,j}}{g_{i,i}}\right]_{|\mathcal{L}|\times|\mathcal{L}|} - I.\end{aligned}$$

Then, the relationship between $\mathbf{p}$ and $\boldsymbol{\gamma}$ if $\{\underline{\gamma}_\ell\}$ are ignored can be described in a matrix form as follows:

$$(I - \mathrm{diag}(\boldsymbol{\gamma})G)\mathbf{p} = \mathrm{diag}(\boldsymbol{\gamma})\tilde{\boldsymbol{\sigma}}. \tag{31}$$

Based on the above equation and according to [19], Problem 4 can be equivalently transformed into

Problem 5 (Transformed Relaxed Slave Problem):

$$\max_{\Gamma=\mathrm{diag}(\boldsymbol{\gamma})} \quad \mathbf{1}' \log(1+\Gamma)\mathbf{w} \tag{32}$$

$$\text{s.t.} \quad \rho\left(\Gamma\tilde{G}_\ell\right) \le 1 \qquad \forall \ell \in \mathcal{L} \tag{33}$$

$$\Gamma C \Gamma = 0 \tag{34}$$

where $\tilde{G}_\ell = G + [1/(\bar{p}_\ell)]\tilde{\boldsymbol{\sigma}}\mathbf{e}'_\ell$. Notice that the constraint (34) is equivalent to (26) since (2) is equivalent to $c_{l,\ell}\gamma_l\gamma_\ell = 0$.

The objective function in the above problem is strictly concave in which the variables $\{\gamma_\ell\}$ are decoupled. We define $\mathbf{y}_{\mathrm{DBN}} = \boldsymbol{\lambda}(\boldsymbol{\gamma}^*)$ and use $\{(\mathbf{w}, \mathbf{y}_{\mathrm{DBN}})\}$ as the training set to train the neural networks, where $\boldsymbol{\gamma}^*$ is the optimal solution of Problem 5. Accordingly, the input and output of the neural networks become $(\mathbf{w}, \boldsymbol{\lambda}(\boldsymbol{\gamma}^*))$.

To obtain the training sets $\{(\mathbf{w}, \mathbf{y})\}$, each time the input $\mathbf{w}$ is uniformly randomly generated within [0, 1] with some of the elements set as 1. Our simulations suggest that in many cases solving Problem 2 in the running phase will generate sparse $\mathbf{w}$'s, i.e., some elements are 0. However, we tried not to use sparse $\mathbf{w}$'s to compute the training set due to the reduction of information the neural networks can learn. In fact, suppose $w_\ell = 0$, the optimal p^*_ℓ is 0 if $\mathbf{w} \ne \mathbf{0}$. In addition, if $\mathbf{w} = \mathbf{0}$, any $\mathbf{p}^*$ will be the optimal solution, which is another reason to avoid using too sparse $\mathbf{w}$'s.

C. Training Phase

In this phase, we train two SVM classifiers and a DBN for each link separately.

1) SVM Classifiers: Let ω_0, ω_1, and ω_2 represent the classes of $\omega_0 = \{y : y \le \delta_{\ell,1}\}$, $\omega_1 = \{y : \delta_{\ell,1} \le y \le \bar{y}_\ell - \delta_{\ell,2}\}$, and $\omega_2 = \{y : y \notin \omega_0, y \notin \omega_1\}$, respectively, where $\delta_{\ell,1}$ and $\delta_{\ell,2}$ are two small threshold numbers. To classify a sample, two SVMs are established and trained. The first SVM decides whether a sample belongs to ω_0 or not. Hence, we label the samples as $z^\ell = 1$ if $y_\ell \in \omega_0$ and $z^\ell = -1$ if otherwise. The output of the SVM can be written as follows:

$$z_0^\ell = \mathrm{sign}\left(\sum_i a_{0,i}^\ell \mathcal{K}(\mathbf{w}_i, \mathbf{w}) + b_0^\ell\right) \tag{35}$$

where i is the index of training samples, $a_{0,i}^\ell$ is a weight, $\mathbf{w}_i$ is the input of sample i, $\mathcal{K}(\cdot,\cdot)$ is the kernel function and b_0^ℓ is the bias. In this paper, the kernel functions are chosen linear for simplicity. Then, y_i corresponding to $\mathbf{w}_i$ is classified to ω_0 if $z_0^\ell = 1$, and not to ω_0 if $z_0^\ell = -1$. In the special case that $z_0^\ell = 0$, y_i can either belong to ω_0 or not. Thus, the training process for finding the maximum-margin hyperplane is equivalent to solving the following minimization problem:

$$\begin{aligned}&\min \quad \left\|\mathbf{a}_0^\ell\right\| \\ &\text{s.t.} \quad z_i^\ell\left(\mathbf{a}_0^{\ell\prime}\mathbf{w} + b_0^\ell\right) \ge 1 - \zeta_i \qquad \forall i\end{aligned}$$

where $\|\cdot\|$ represents the norm of a vector. $\mathbf{a}_0^\ell$ is the vector formed by $\{a_{0,i}^\ell\}$. $\zeta_i \ge 0$ is a constant to tolerate classification faults. The SVM for deciding whether a sample lies in ω_2 or not is similarly trained.

2) DBNs: We establish for each link a DBN, denoted as DBN_ℓ, to provide a refined approximate of the optimal power allocation, as shown in Fig. 3. For each RBM, let $\mathbf{v}$ and $\mathbf{h}$ denote the vectors of the visible and hidden neurons, respectively. The unsupervised training in the first step is to initialize the parameters, including the weights (denoted as matrix Φ) between the two visible and hidden layers, the biases (denoted as vectors $\mathbf{b}_v$ and $\mathbf{b}_h$) associated with the visible and hidden neurons, respectively, at an optimal configuration. Let the parameters be $\varphi = (\Phi, \mathbf{b}_v, \mathbf{b}_h)$. Then, the parameters are updated iteratively as follows:

$$\varphi_{t+1} = \varphi_t + \eta \frac{\partial \log \Pr(\mathbf{v}_t)}{\partial \varphi} \tag{36}$$

where η is the learning rate. $\Pr(\mathbf{v})$ is the probability of $\mathbf{v}$ (one of the training data), which can be given as

$$\Pr(\mathbf{v}) = \sum_{\mathbf{h}} \Pr(\mathbf{v}, \mathbf{h}) = \sum_{\mathbf{h}} \frac{\exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}}\sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}. \tag{37}$$

In the above, $\Pr(\mathbf{v}, \mathbf{h})$ is the joint distribution of the visible and hidden layers. $E(\mathbf{v}, \mathbf{h})$ represents the energy function associated with the RBM and takes the form

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}'\Phi\mathbf{h} - \mathbf{b}'_v\mathbf{v} - \mathbf{b}'_h\mathbf{h}. \tag{38}$$

Notice that it is of great computation complexity to calculate $\Pr(\mathbf{v}, \mathbf{h})$ as in (37). To this end, the contrastive divergence method as proposed in [32], which approximates $\Pr(\mathbf{v}, \mathbf{h})$ based on Gibbs sampling, may be applied.

The supervised training process is to fine tune the parameters with the output y_ℓ of sample data by minimizing a cost function called the cross entropy

$$S = -\frac{1}{D}\sum_{i=1}^{D}\left(y_\ell^{(i)} \log\left(\hat{y}_\ell^{(i)}\right) + \left(1 - y_\ell^{(i)}\right)\log\left(1 - \hat{y}_\ell^{(i)}\right)\right) \tag{39}$$

where $y_\ell^{(i)}$ and $\hat{y}_\ell^{(i)}$ are the output of the ith training data and that of the predicted output by the DBN$_\ell$ given the input of the ith training data, respectively. In fact, S is a measure of the prediction accuracy of DBN$_\ell$. Then, by the backpropagation method, the parameters are tuned iteratively based on the gradient method as follows:

$$\varphi_{t+1} = \varphi_t - \tilde{\eta}\frac{\partial S}{\partial \varphi} \tag{40}$$

where $\tilde{\eta}$ is the learning rate in the gradient method.

D. Running Phase

In the running phase, given an input $\mathbf{w}$ (after normalization) from the solution to the master problem, the SVMs are first applied to perform classification for each link. Specifically, for each link ℓ, if it belongs to class ω_0 (resp. ω_2), its power is set as $p_\ell = 0$ w.p. (with probability) $\phi_{0,\ell}$ (resp. $p_\ell = \bar{p}_\ell$ w.p. $\phi_{2,\ell}$), where $\phi_{0,\ell}$ and $\phi_{2,\ell}$ indicate the accuracies of the two SVMs, respectively, i.e., they are the probabilities that the SVMs correctly classify the inputs of the training data. If a link is categorized to class ω_1 or its power is not set 0 nor $\bar{p}_\ell$ in the above, the DBN is applied to further decide its power. Specifically, with the input $\mathbf{w}$, DBN$_\ell$ outputs $\mathbf{y}_\ell$. Then, the corresponding solution to Problem 4 can be computed based on (31) by

$$\mathbf{p} = \left(\frac{1}{\xi}I - \tilde{\mathbf{Y}}G\right)^{-1}\tilde{\mathbf{Y}}\tilde{\boldsymbol{\sigma}} \tag{41}$$

where $\tilde{\mathbf{Y}} = \mathrm{diag}(\exp(\mathbf{y})) - I$. ξ is a scaling factor that adjusts the solution $\mathbf{p}^*$ to ensure that each solution p_ℓ is nonnegative. Here, we assume that $\sum_{j\in\mathcal{L}\setminus i} g_{j,i} > 0$, $\forall i \in \mathcal{L}$. In other words, each link has at least one interfering link, thus avoiding the trivial problem of allocating power to those isolated links. With this assumption, we can see that $\rho(G) < 1$. Due to the above problem relaxation and approximation, the output of the neural networks may be infeasible, resulting in that $\rho(\tilde{\mathbf{Y}}G) > 1$. In this case, we can tune ξ such that $\xi\rho(\tilde{\mathbf{Y}}G) \leq 1$ and that the above equation has a feasible solution.

In view of (3), a too small p_ℓ is unneeded. Therefore, after obtaining $\mathbf{p}$, for those links that have $\gamma_\ell < \underline{\gamma}_\ell$, we set the corresponding power $p_\ell = 0$. Such modifications will keep the capacity of those links remaining 0 and may increase the capacity of others (due to reduction in interference).

The whole procedure that iteratively solves Problems 2 and 3 is summarized in Algorithm 1. In each step k, the dual values obtained by solving Problem 2 is first normalized and input to the classifiers and neural networks to predict the outputs $\mathbf{y}_{\mathrm{SVM}}$ and $\mathbf{y}_{\mathrm{DBN}}$. The SVM classifiers are used to decide whether to assign maximum power or to turn off a link at first, since these two cases happen most frequently. At the same time, the DBNs take the input and predict $\mathbf{y}$ by which the corresponding $\mathbf{p}$ is calculated. Then, we calculate $\boldsymbol{\lambda}(\mathbf{p})$ where the sensitivity level $\underline{\gamma}_\ell$ is applied to set $p_\ell = 0$ if necessary. In addition, in order to avoid the infeasible power allocation that conflicting links are simultaneously activated, we apply a simple greedy

Algorithm 1: MLA for Power Control

```
input     : Flow demands $\{\mathbf{q}_1, \ldots, \mathbf{q}_{|\mathcal{F}|}\}$
initialize: Initial allocation set $P$
while $true$ do
    solve the linear program: Problem 2;
    obtain the (normalized) weights $\mathbf{w}$;
    for $\ell \in \mathcal{L}$ do
        use $SVM_\ell$ to do classification;
        use $DBN_\ell$ to predict $y_\ell$;
    end
    calculate $\mathbf{p}$ based on (41);
    for $\ell \in \mathcal{L}$ do
        //set $p_\ell$ based on the classification results:
        if $\omega_0$ then
            $p_\ell \leftarrow 0$ w.p. $\phi_{0,\ell}$;
        else if $\omega_2$ then
            $p_\ell \leftarrow \bar{p}_\ell$ w.p. $\phi_{2,\ell}$;
        end
    end
    calculate $\boldsymbol{\lambda}(\mathbf{p})$;
    if $\lambda_\ell < \underline{\gamma}_\ell$ then
        $p_\ell \leftarrow 0$;
    end
    //address the conflicting links issue
    $\mathcal{L}_c \leftarrow \{\ell : p_\ell > 0\}$;
    while $\mathcal{L}_c \neq \emptyset$ do
        $l \leftarrow \arg\max_\ell \lambda_\ell w_\ell$;
        for $i \in \mathcal{L}_c$ and $i \neq l$ do
            if $c_{i,l} \neq 0$ then
                $p_i \leftarrow 0$; $\mathcal{L}_c \leftarrow \mathcal{L}_c\setminus\{i\}$;
            end
            $\mathcal{L}_c \leftarrow \mathcal{L}_c\setminus\{l\}$;
        end
    end
    if $\mathbf{w}'\boldsymbol{\lambda}(\mathbf{p}) < \epsilon$ then
        algorithm stops;
    end
    add column $[\mathbf{0}_{|\mathcal{N}||\mathcal{F}|}, \boldsymbol{\lambda}(\mathbf{p}), 1]'$ to the right side of $P$;
end
```

algorithm to iteratively address this issue.[1] As shown in lines 22–31 in Algorithm 1, in each iteration, the activated link with the largest $\lambda_\ell w_\ell$ is chosen and all the other links that conflict with this selected link will be deactivated by setting their transmit power to 0. In the end of each iteration, the vector $\boldsymbol{\lambda}(\mathbf{p})$ is used to construct a new column which is padded to the right side of the matrix A to update Problem 2. The algorithm stops if $\mathbf{w} = 0$ or $\mathbf{w}'\boldsymbol{\lambda}(\mathbf{p})$ is smaller than a certain threshold ϵ. That is, no new column can be found that yields a reduced cost as in (23) higher than some small value.

VI. Simulation Results

In this section, we present simulation evaluations of the proposed method. As shown in Fig. 5, we consider a 16-node wireless network deployed in a 800×600 m^2. The maximal communication range is set as 200 m, resulting in 26 undirected links (or totally 52 directed links). The interference range is 250 m. The channel gain $g_{\ell',\ell}$ is set inversely proportional to the square distance between Tx(ℓ') and Rv(ℓ).

[1]Note that this greedy algorithm is different from that in [27].

TABLE II
FOUR SIMULATION CASES

Case	# flows	Source-destination pairs $\{(s_f, d_f)\}$	Flow demand $\{q_f\}$	Link conflict
1	2	$\{(9,13),(5,15)\}$	$\{0.5, 1.5\}$	low
2	3	$\{(11,3),(1,12),(16,9)\}$	$\{1,1,2\}$	moderate
3	3	$\{(5,3),(8,14),(6,15)\}$	$\{0.4, 0.5, 1.5\}$	high
4	5	$\{(1,2),(9,10),(6,7),(4,5),(14,10)\}$	$\{2, 0.2, 1.5, 1.5, 0.5\}$	free

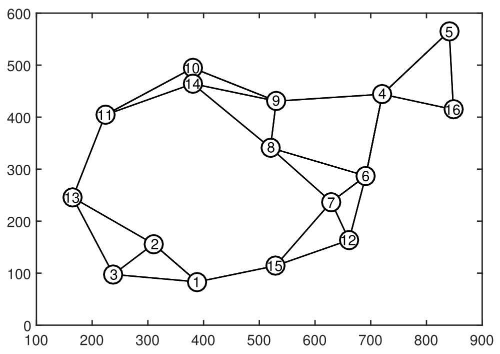
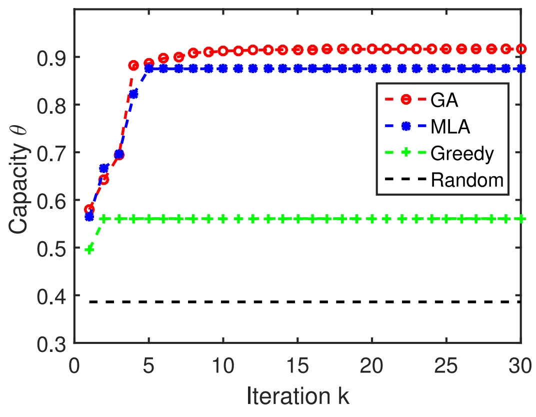
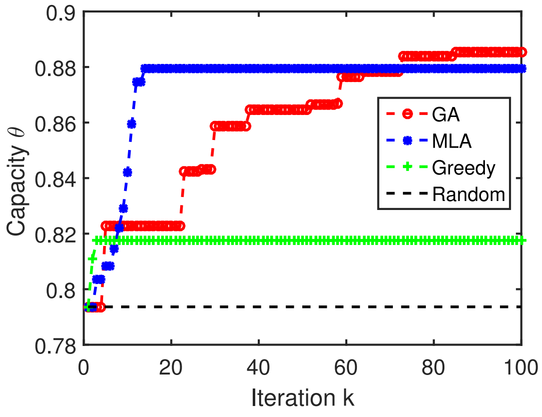


Fig. 5. Grid network topology.



(a)



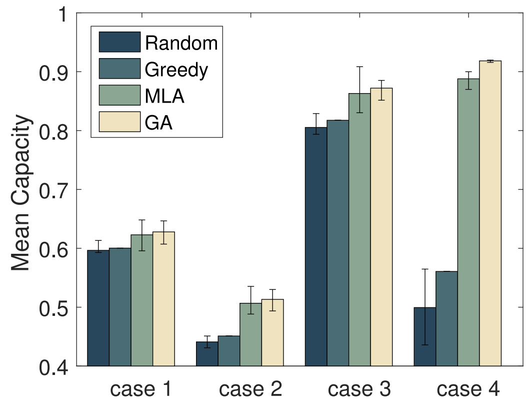
(b)

Fig. 6. Performance comparison of single runs. (a) Without conflict. (b) With conflict.

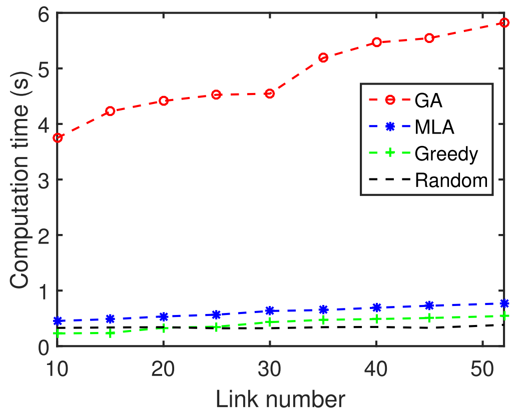
The maximum transmit power of each link is $\bar{p}_\ell = 1$ unit. The bandwidth of the links is set as $B = 1$. All the nodes can serve as relays for forwarding data until their destinations. The simulation program is built within MATLAB, where we use deep neural network codes [33] and LibLinear library [34] for constructing, training, and applying the DBNs and SVMs in our algorithm, respectively. The simulations are conducted on a Dell PowerEdge T630 computer with two Intel E5-2620 CPUs and 16GB RAM.

Each DBN$_\ell$ contains four layers with the input, two hidden and the output layers, respectively, contain 52, 60, 5, and 1 neurons. We generate random $\mathbf{w}$'s and solve the slave problem optimally using GA to generate sample solutions. Twenty thousand samples are used to train the learning systems. In the running phase, we compare the performance of our proposed algorithm MLA with that of three other algorithms: 1) the GA which iteratively searches the optimal solutions; 2) the greedy algorithm (Greedy) which greedily solves the slave problem [27]; and 3) a random algorithm (Random), when they are applied to solve the slave problem. Random randomly outputs an allocation vector $\mathbf{p}$ regardless of the input $\mathbf{w}$ (conflicting links are pruned by the greedy algorithm similar to that in Algorithm 1).

Fig. 6 shows the performance comparison results among the four algorithms, where we consider two scenarios, namely with and without conflict. For the scenario without conflict, we consider a reduced network with 5 (directed) links selected from the original network and these links that are mutually conflict free, i.e., $c_{l,l'} = 0$ for any pair of the links. Such a reduced network is disconnected; however, the different parts of the network may have mutual interference. Five source-destination pairs [i.e., (1, 2), (9, 10), (6, 7), (4, 5), and (14, 10)] are chosen with the flow demands as 2, 0.2, 1.5, 1.5, and 0.5, respectively, which corresponds to Case 4 in Table II. In this case, each DBN$_\ell$ contains four layers with the input, two hidden and the output layers, respectively, contain 5, 20, 4, and 1 neurons. The simulation results in Fig. 6(a) show that the proposed MLA achieves very close performance to the GA, indicating its near-optimal performance. Besides, MLA obviously outperforms Greedy in terms of achieved capacity. For the scenario with conflict, we consider the original 52-link network and focus on source-destination pairs (5, 3), (8, 14), and (6, 15) with the corresponding demands 0.4, 0.5, and 1.5, respectively, which corresponds to Case 3 in Table II. Fig. 6(b) shows similar comparison results as Fig. 6(a) except that GA converges slower than MLA since new power allocations are continuously found. Greedy converges quicker than MLA roughly because that greedily solving the slave problem will activate only a few links, quickly leading to the situation that no new power allocations can be found. In both figures, the Random algorithm achieves the worst performance since it finds new power allocations planlessly.



(a)



(b)

Fig. 7. Performance comparison in different network scenarios. (a) Mean achieved capacity. (b) Average per-step computation time.

We further conduct simulations under four cases of network settings as shown in Table II. These cases have different levels of conflicts among the links. For each case, we conduct 50 independent simulation runs, and display in Fig. 7(a) the mean achieved capacity with 95% confidence intervals, where for each run the capacity is calculated as that achieved after 100 iterations. We can observe that MLA always achieves close performance to GA, and obviously outperforms Greedy and Random in all the four cases. In each case, the performance of Greedy almost does not change in different simulation runs; however, the performance of all the other three varies because they incorporate probabilistic operations. Particularly, MLA probabilistically accepts the classification results of SVMs, offering more opportunity to finding power allocations. Although MLA is inferior over GA in terms of achieved capacity, its computation time is much lower than that of GA as shown in Fig. 7(b). The two figures also tell that, at similar computing costs, MLA achieves obviously higher capacity than Greedy.

VII. Conclusion

We have studied the problem of joint flow allocation, link scheduling, and power control for capacity optimization in wireless networks. We decomposed the original problem into a linear program and a nonlinear one, and proposed an MLA integrated with both SVM and deep belief neural network techniques to solve the nonlinear subproblem. Simulation results demonstrate that the proposed algorithm is able to achieve a near-optimal network capacity at a cost much lower than the GA and similar to that of the simple greedy algorithm. This paper demonstrates the potential advantages of machine learning algorithms in solving network resource allocation problems. In future work, we will delve into the learning algorithms and design neural networks adapted to wireless networks.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] W. Xu *et al.*, "Internet of Vehicles in big data era," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018.
[3] Z. Sun, L. Zhou, and W. Wang, "Learning time-frequency analysis in wireless sensor networks," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2017.2771514.
[4] L. Xue, C. Sun, D. Wunsch, Y. Zhou, and F. Yu, "An adaptive strategy via reinforcement learning for the prisoner's dilemma game," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 1, pp. 301–310, Jan. 2018.
[5] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.
[6] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.
[7] X. Cheng, L. Fang, L. Yang, and S. Cui, "Mobile big data: The fuel for data-driven wireless," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1489–1516, Oct. 2017.
[8] B. Mao *et al.*, "Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1946–1960, Nov. 2017.
[9] X. Cheng, L. Fang, X. Hong, and L. Yang, "Exploiting mobile big data: Sources, features, and applications," *IEEE Netw.*, vol. 31, no. 1, pp. 72–79, Jan./Feb. 2017.
[10] X. Cao, L. Liu, Y. Cheng, and X. S. Shen, "Towards energy-efficient wireless networking in the big data era: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 303–332, 1st Quart., 2018.
[11] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.
[12] M. H. Kim and M.-G. Park, "Bayesian statistical modeling of system energy saving effectiveness for MAC protocols of wireless sensor networks," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, R. Lee and N. Ishii, Eds. Berlin, Germany: Springer, 2009, pp. 233–245.
[13] B. Bojović, E. Meshkova, N. Baldo, J. Riihijärvi, and M. Petrova, "Machine learning-based dynamic frequency and bandwidth allocation in self-organized LTE dense small cell deployments," *EURASIP J. Wireless Commun. Netw.*, vol. 2016, no. 1, p. 183, 2016.
[14] P. Wang, S.-C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, 2016, pp. 760–765.
[15] N. Duffield, C. Lund, and M. Thorup, "Estimating flow distributions from sampled flow statistics," in *Proc. ACM Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2003, pp. 325–336.
[16] L. Ghouti, "Mobility prediction using fully-complex extreme learning machines," in *Proc. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn.*, 2014, pp. 607–612.
[17] L. Liu, Y. Cheng, L. Cai, S. Zhou, and Z. Niu, "Deep learning based optimization in wireless network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.
[18] L. Liu, B. Yin, S. Zhang, X. Cao, and Y. Cheng, "Deep learning meets wireless network optimization: Identify critical links," *IEEE Trans. Netw. Sci. Eng.*, to be published, doi: 10.1109/TNSE.2018.2827997.
[19] C. W. Tan, S. Friedland, and S. H. Low, "Nonnegative matrix inequalities and their application to nonconvex power control optimization," *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 3, pp. 1030–1055, 2011.

[20] E. Anderson, C. Phillips, D. Sicker, and D. Grunwald, “Optimization decomposition for scheduling and system configuration in wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 271–284, Feb. 2014.

[21] L. P. Qian, Y. J. Zhang, and J. Huang, “MAPEL: Achieving global optimality for a non-convex wireless power control problem,” *IEEE Trans. Wireless Commun.*, vol. 8, no. 3, pp. 1553–1563, Mar. 2009.

[22] C. W. Tan, M. Chiang, and R. Srikant, “Fast algorithms and performance bounds for sum rate maximization in wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 706–719, Jun. 2013.

[23] Y. Wu and D. H. K. Tsang, “Distributed power allocation algorithm for spectrum sharing cognitive radio networks with QoS guarantee,” in *Proc. IEEE INFOCOM*, 2009, pp. 981–989.

[24] J. Luo, C. Rosenberg, and A. Girard, “Engineering wireless mesh networks: Joint scheduling, routing, power control, and rate adaptation,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1387–1400, Oct. 2010.

[25] L. Fu, S. C. Liew, and J. Huang, “Fast algorithms for joint power control and scheduling in wireless networks,” *IEEE Trans. Wireless Commun.*, vol. 9, no. 3, pp. 1186–1197, Mar. 2010.

[26] Y. Wei, F. R. Yu, M. Song, and Z. Han, “User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach,” *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Jan. 2018.

[27] 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.

[28] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6. Belmont, MA, USA: Athena Sci., 1997.

[29] L. Liu *et al.*, “Energy-efficient capacity optimization in wireless networks,” in *Proc. IEEE INFOCOM*, 2014, pp. 1384–1392.

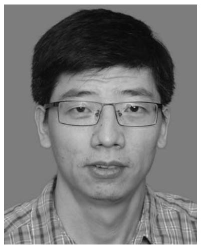
[30] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[31] G. E. Hinton, “Deep belief networks,” *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[32] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[33] M. Tanaka and M. Okutomi, “A novel inference of a restricted Boltzmann machine,” in *Proc. IEEE Int. Conf. Pattern Recognit. (ICPR)*, 2014, pp. 1526–1531.

[34] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LibLinear: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jan. 2008.



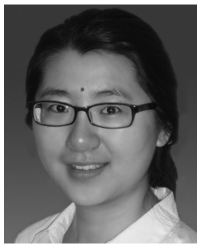
Xianghui Cao (S’08–M’11–SM’16) received the B.S. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively.

From 2012 to 2015, he was a Senior Research Associate with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. He is currently an Associate Professor with the School of Automation, Southeast University, Nanjing, China. His current research interests include cyber-physical systems, wireless network performance analysis, wireless networked control, and network security.

Dr. Cao was a recipient of the Best Paper Runner-Up Award of ACM MobiHoc14. He also serves as an Associate Editor for several journals, including *Acta Automatica Sinica*, *IEEE/CAA Journal of Automatica Sinica*, *KSII Transactions on Internet and Information Systems*, *Security and Communication Networks*, and *International Journal of Ad Hoc and Ubiquitous Computing*. He served as the Publicity Co-Chair for ACM MobiHoc15, the Symposium Co-Chair for ICNC17 and IEEE/CIC ICCC15, and has been a TPC member for a number of conferences.

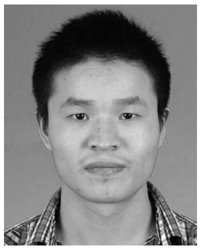
Rui Ma received the B.S. degree in automation engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2016. She is currently pursuing the M.S. degree in control science and engineering at Southeast University, Nanjing.

Her current research interests include machine learning and network scheduling.



Lu Liu (S’13–M’18) received the Ph.D. degree in computer engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 2017.

Her current research interest includes energy efficient networking and communication, machine learning-based network optimization, and fog and edge computing for the Internet of Things.



Hongbao Shi received the B.S. degree in automation from the Hefei University of Technology, Hefei, China, in 2015, and the M.E. degree in control science and engineering from Southeast University, Nanjing, China, in 2018.

His current research interests include cyber-physical system security and network resource allocation.

Yu Cheng (S’01–M’04–SM’09) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

He is currently a Full Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His current research interests include wireless network performance analysis, network security, big data, cloud computing, and machine learning.

Dr. Cheng was a recipient of the Best Paper Award of QShine 2007, IEEE ICC 2011, the Runner-Up Best Paper Award of ACM MobiHoc 2014, the National Science Foundation CAREER Award in 2011, and the IIT Sigma Xi Research Award in the Junior Faculty Division in 2013. He is an Associate Editor of the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He has served as the Symposium Co-Chair for IEEE ICC and IEEE GLOBECOM and the Technical Program Committee Co-Chair for WASA 2011 and ICNC 2015. He is the Founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc Distinguished Lecturer in 2016 to 2017.

Changyin Sun received the B.S. degree from the Department of Mathematics, Sichuan University, Chengdu, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Southeast University, Nanjing, China, in 2001 and 2004, respectively.

He is currently a Full Professor with the School of Automation, Southeast University. His current research interests include intelligent control, neural networks, data-driven control, and optimization algorithms.

Dr. Sun was a recipient of the National Science Fund for Distinguished Young Scholars of China. He was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Neural Processing Letters*, and *IEEE/CAA Journal of Automatica Sinica*. He is a Fellow of the Chinese Association of Automation.