




A Self-Supervised Learning Approach for Accelerating Wireless Network Optimization

Shuai Zhang , *Student Member, IEEE*, Oluwaseun T. Ajayi , *Student Member, IEEE*,
and Yu Cheng , *Senior Member, IEEE*

Abstract—The prevailing issue in multi-hop wireless networking is interference management, which militates against the efficiency of traditional routing and scheduling algorithms. We develop a self-supervised learning approach to address the classic NP-hard problem of capacity optimization over a multi-hop wireless network, where the routing and scheduling decisions are deeply coupled. Our two-stage design leverages historical computation experiences to accelerate the optimization of new problem instances, where an instance represents the application-level input containing network topology, interference model, and other user-level traffic constraints. The first stage, Scheduling Structure Classification (SSC), distills the scheduling structure of the historical optimization instances into an appropriate number of classes, through a properly designed clustering algorithm without prior assumption or knowledge. The second stage uses the instances labelled with class information from the first stage to train an Application Identification (AID) neural network model capable of predicting a future problem instance's scheduling class given its application-level information. When solving the new instance, its predicted scheduling class and the associated scheduling structure are exploited to compute an efficient approximate solution, avoiding the time-consuming iterative search for such scheduling structure as practiced by the conventional approaches. We apply our method to different types of wireless multi-commodity flow problems across various network sizes and disparate flow requirements. The results demonstrate that our method significantly reduces the computation time robustly by at least 70% with only slight loss in the solution quality.

Index Terms—Deep learning, wireless network optimization, topology representation.

I. INTRODUCTION

EMERGING mobile applications, such as vehicular ad hoc networks, industrial Internet of Things, and Unmanned Aerial Vehicles networks, are calling people's attention to reexamine the challenging classic problem of capacity optimization over a multi-hop wireless network. Despite long-term efforts in this area, the efficient optimization of large-scale wireless networks remains a challenging task. This is largely due to the

fact that many of these tasks can be reduced to NP-hard problems like independent set problems, graph coloring, and even their approximation is shown to be infeasible [1], [2].

While the conventional wireless network optimization algorithms have not gained much recent progress, we resort to an innovative data-driven, learning-based approach to significantly reduce the computational overhead in wireless network optimization as well as maintaining a close-to-optimum performance. In this paper, we specifically study the classic wireless network flow problem, where the network-layer routing issue and link-layer scheduling issue are coupled together [3]. It is worth noting that although there are some recent machine-learning based studies in the context of wireless networks, they focus on either wired networking or scheduling in the single-hop networking scenarios [4], [5].

Our study is inspired by the idea of extracting knowledge from historical problem instances to accelerate the solution of new instances, proposed in recent works [6], [7]. The goal is that when a new problem instance is detected to be similar to another one whose solution was obtained at a previous time, certain information from that historical case can be leveraged to facilitate the solution process of this new instance. Although this idea is intuitive, its implementation is by no means trivial. To achieve this goal, two fundamental challenges need to be addressed.

The first one is how to effectively identify the similarity between two problem instances. The similarity evaluation will not be as simple as directly observing and comparing the application-level problem input information (e.g., the network topology or the commodity flow deployment), because a slight difference in these input items can cause significant changes to problem outputs, rendering the historical instance's solution information less useful. For example, a new instance may deploy the same number of commodity flows as that in a historical instance; however, if one flow involves a different source or destination node in the new instance, the scheduling might be largely different from the historical one. The other challenge is the proper usage of the historical instances. Even with a means to identify the similarity between problem instances, directly applying the old instances' solutions is unlikely to give good performance, neither is it practical to store all the past solutions due to the huge size of input configuration space. Our vision is that the historical computation experiences must be distilled and used in an intelligent way to facilitate solving new instances without rote-memorization of all experienced instances.

Manuscript received 2 July 2022; revised 28 October 2022 and 20 January 2023; accepted 28 January 2023. Date of publication 10 February 2023; date of current version 20 June 2023. This work was supported in part by the National Science Foundation (NSF) under Grants CNS-1816908 and CNS-2008092. The review of this article was coordinated by Prof. Hongzi Zhu. (*Corresponding author: Yu Cheng.*)

The authors are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: szhang104@hawk.iit.edu; oajayi6@hawk.iit.edu; cheng@iit.edu).

Digital Object Identifier 10.1109/TVT.2023.3244043

This paper addresses these two challenges with an approach that follows the self-supervised learning paradigm. Our method does not rely on human knowledge or pre-existing labels to identify similarity between instances. It consists of two stages.

The first stage is *Scheduling Structure Classification* (SSC). Optimization tasks in wireless networks can be generically formulated as scheduling link subsets that do not interfere with each other over the same communication channel. Such subsets are termed *independent sets* (IS) as they correspond to the nodes not connected by arcs on conflict graphs representing the links' interference relationship [8]. We will demonstrate that the scheduled independent sets (also called the scheduling structure) associated with each optimization instance offer a suitable representation of the internal structure for evaluating the similarity between problem instances with different application-level inputs. We develop a clustering algorithm to group the training instances by their scheduled ISs into an appropriate number of *scheduling classes*, each representing problem instances that have the similar scheduling structure. It is worth noting that the number of such classes is much smaller than the number of training instances. The clustering outcome can be interpreted as a scalable extract of the historical computing experiences.

The output from the first stage offers labeled data to train the second stage, the *Application Identification* (AID) model, in a supervised learning manner. The AID model is capable of predicting a problem instance's class given its application-level information, such as the network topology and commodity flow requirement. When solving a new problem instance, the schedulable independent sets suggested by the AID model will be directly exploited to compute an approximate solution, obtained through one-round of linear programming (LP), avoiding the time-consuming search for such scheduling structure information adopted in the conventional algorithms.

With the increasingly wide deployment of emerging applications such as wireless mesh network [9], [10], [11], space-air-ground integrated networks [12], [13], and 5G/6G integrated access and backhaul (IAB) systems [14], [15], our method can be readily applied in such scenarios, where the past wireless link interference between the base stations with backhauling needs can be deduced from data to suggest high-quality independent sets with less time. For this reason, we apply our method to the multi-commodity flow optimization problem in wireless networks under link interference constraints. We demonstrate through extensive experiments with different-sized networks and flow requirements that our method significantly reduces the computation time by at least 70%, and it does so robustly with little loss in solution quality. Then to further show that the method is effective not just with one type of specific flow problem, we test the performance under a different flow optimization scenario and have found similarly consistent performance improvements.

The remainder of this paper is organized as follows. We first formulate the wireless network flow optimization problem to be solved. We then present all the design and implementation details of the self-supervised learning based approach. Next comes the numerical results, followed by a review of more related work and the conclusion remarks. The main notations used in this paper together with the list of acronyms are summarized in Table I.

TABLE I
LIST OF SYMBOLS AND ACRONYMS

Symbol	Description
\mathcal{N}, \mathcal{L}	node set, link set
$f_d(u, v)$	amount of flow d on a link
r_d	total amount of a commodity flow d
w_d	weight of a commodity flow d
\mathcal{M}	set of schedulable ISs
α_m	transmission time of a scheduled IS m
$p_m(u, v)$	capacity of a link in a scheduled IS m
Acronym	Meaning
AID	Application Identification
AR	Approximation Ratio
OBCR	Optimal Bit Complexity Randomized method
CIS	Critical Independent Sets
CTR	Computation Time Reduction
DCG	Delayed Column Generation
EVR	Explained Variance Ratio
IS	Independent Sets
LP	Linear Programming
MCF	Multi-commodity Flow
ML	Machine Learning
PCs	Principal Components
SSC	Scheduling Structure Classification

II. WIRELESS NETWORK FLOW OPTIMIZATION

We consider a generic wireless network $(\mathcal{N}, \mathcal{L})$, with a node set \mathcal{N} and a link set \mathcal{L} . Each node has attributes *communication range* and *interference range*, representing the maximum distances two nodes must be at to communicate or interfere with each other. There exists a directed link $(u, v) \in \mathcal{L}$ from a transmitter node u to a receiver node v if and only their Euclidean distance is smaller than or equal to the communication range of node v .

The *protocol interference* model is used. It stipulates that two links interfere if the transmitter node of one link is within the interference range of the receiver node of another link. In this paper, we assume that all nodes of the network have identical communication and interference ranges, and that the interference range is greater than the communication range. We only consider link schedules that are interference-free, and for convenience, we use *independent set* (IS) to denote a set of network links that do not interfere with each other when they transmit at the same time. The link capacity with no interference $c(u, v)$ is calculated by the Shannon formula, as a function of the distance between u and v , transmission power and noise power which are given as known parameters.

A. Multi-Commodity Flow Optimization

Within the given wireless network, traffic demands

$$[(\text{src}_0, \text{dst}_0), \dots, (\text{src}_{D-1}, \text{dst}_{D-1})]$$

refer to the list of D pairs of nodes that need to have messages sent from the source to the destination. Each traffic flow associated with one pair is referred to as a *commodity flow*. The nodes within these pairs are usually not directly reachable by one wireless link, so the network traffic may go through multiple hops, which would further depend on the link scheduling considering

their interference relationships. We use \mathcal{N}_u to refer to the set of nodes that are neighbors (reachable by a direct link) of node u , and can be further superscripted by $-$ or $+$ to differentiate the “in-neighbors” and the “out-neighbors,” i.e., neighbor nodes connected by a link to u and from u , respectively. Let $f_d(u, v)$ denote the amount of flow carried over link (u, v) for commodity d . The total flow value r_d for commodity d is the sum of all link flows out of the source node:

$$r_d = \sum_{v \in \mathcal{N}_{\text{src}_d}^+} f_d(\text{src}_d, v) \quad (1)$$

Our objective is to optimize the network utility as a function of the achieved commodity flows. For example, it could be the weighted sum of the commodity flows: $\sum_d w_d r_d$, where the per-commodity flow weight coefficient w_d is given as part of the problem input, to signify its own service priority. The optimization is subject to the following constraints.

a) Flow conservation constraints: For each network node that is neither the source nor the destination of a commodity flow, the net amount of traffic flows should be zero. Note that this should hold for each node and for each commodity: the amount of traffic flow into the node should be equal to that of the flow out of the node.

$$\sum_{u \in \mathcal{N}_v^-} f_d(u, v) = \sum_{u' \in \mathcal{N}_v^+} f_d(v, u'), \quad \forall v \neq \text{src}_d, \text{dst}_d; \forall d. \quad (2)$$

Note that the flow conservation constraints implicitly determine the routing of each commodity flow.

b) Link capacity constraints: Suppose that all schedulable ISs are contained in set \mathcal{M} , and an IS m gets scheduled a time share α_m . The effective capacity $p_m(u, v)$ of link (u, v) when IS m is active takes value $c(u, v)$ if such link is part of the IS m , and 0 otherwise.

Then the constraints can be defined such that for each link, the sum of flows belonging to all commodities should not exceed its achievable capacity under the current scheduling. Because the system works in a time-multiplexing way, the link’s achievable capacity is the weighted sum of the effective capacity, i.e., weighted by the time portion it is active.

$$\sum_{d=0}^{D-1} f_d(u, v) \leq \sum_{m \in \mathcal{M}} \alpha_m p_m(u, v), \quad \forall (u, v) \in \mathcal{L},$$

$$p_m(u, v) = \begin{cases} c(u, v) & \text{if } (u, v) \text{ is active in } m \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

c) Scheduling constraint: Recall that if one forms a *conflict graph* whose vertices correspond to the network links and each of its arcs represents the interference relationship between a pair of the network links, then a subset of non-interfering network links correspond to the independent set, i.e., a subset of vertices not having an arc between any pair, on the conflict graph. To ensure interference-free transmission, at any given moment, only an IS can be activated, and different ISs are scheduled for transmission in a time-sharing manner to satisfy the flow demands over time.

If the network has $L (= |\mathcal{L}|)$ links, each IS can be represented by an L -dimensional binary vector, where each component’s value taking either 1 or 0 indicates a corresponding link’s active or inactive status. Specifically, let α_m denote the normalized transmission time allocated to IS m , and the time assignment to all ISs must sum to 1:

$$\sum_{m \in \mathcal{M}} \alpha_m = 1 \quad (4)$$

Subject to all the above constraints, a typical multi-commodity flow (MCF) problem of the *weighted sum maximization* type can be formulated as:

$$\text{Maximize}_{\{f_d(u, v)\}, \{\alpha_m\}} \sum_{d=0}^{D-1} w_d r_d$$

s.t. constraints(2), (3), (4),

$$f_d(u, v) \geq 0, \quad \forall (u, v) \in \mathcal{L}, \forall d$$

$$\alpha_m \geq 0, \quad m \in \mathcal{M} \quad (5)$$

The problem described above has the form of linear programming because the objective and constraints are linear functions. However, the size of \mathcal{M} is exponentially large and cannot be easily enumerated; and finding sets of non-interfering links is equivalent to finding a graph coloring of graph edges. Therefore, the problem is essentially a NP-hard problem.

B. Delayed Column Generation

Credited to its LP format, the optimization problem (5) can be efficiently solved approximately by delayed column generation (DCG) method [16]. It is essentially an iterative search method: starting from an initial set of columns in the constraint matrix, each round of the algorithm consists of the solving of a restricted master problem and a sub-problem. The restricted master problem is the same as the main problem except that the constraint matrix only consists of the set of known columns so far. Its dual solution is used as an input to form a sub-problem, whose purpose is to search for a new constraint column that can improve the master problem solution. The new column is then added to the set of known columns and the same iteration continues until a termination condition is met. As the number of iteration grows, the solution of the restricted master problem converges to the optimum value.

The DCG matters in our context, as search for a new column can be equivalent to searching a schedulable IS [17]. Note that with DCG, the sub-problem is an NP-hard maximum weighted independent set problem but can be approximatedly solved with a greedy solution. We implement the DCG algorithm to compute training cases and it also serves as the benchmark method to evaluate the solution quality and time efficiency of our machine learning (ML) based approach.

III. SELF-SUPERVISED LEARNING BASED OPTIMIZATION

Fundamentally, achieving the optimal commodity flow is boiled down to searching the right collection of independent

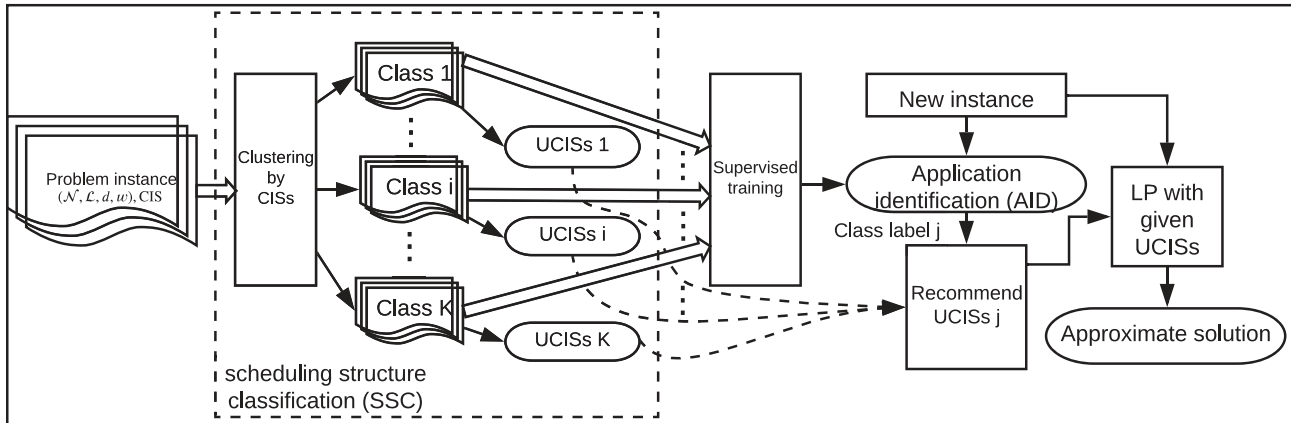


Fig. 1. The self-supervised learning based optimization framework.

sets. Interestingly, the size of scheduled independent sets is a small number and has an upper bound determined by the problem dimension. Specifically, it is shown that the size of the scheduled independent sets is upper-bounded by the rank of the constraint matrix of such wireless network flow optimization problems [18]; using the notation in the previous section, this upper bound would be on the order of $\mathcal{O}(D|\mathcal{N}| + |\mathcal{L}|)$.

This collection of the scheduled ISs corresponding to an optimum flow is hereafter referred to as the *critical independent sets* (CIS). Hence, the essence of the hardness of wireless network flow optimization is to search the exponentially many possible (up to $2^{|\mathcal{L}|}$) independent sets to find the small CIS, whose size is a linear function of $|\mathcal{L}|$.

From the computing perspective, an optimization algorithm can be interpreted as a mapping function, which takes the given application-level inputs (e.g., commodity deployment, network topology, interference model, and other user-level traffic constraints) and outputs the scheduling decisions and related flow allocation. If we are to train a model which is able to approximate such an input-output mapping: given an optimization instance with application-level inputs, it will suggest sets of non-interfering links $\mathcal{M}' \approx \text{CISs}$ which are reasonably close to the CISs used by the optimum scheduling, then we may approximate the original optimization problem by replacing the constraint (4) as

$$\sum_{m \in \mathcal{M}'} \alpha_m = 1. \quad (6)$$

The solution to this simplified linear programming problem can approximate the solution to the original NP-hard problem. No doubt, how close this solution is to the optimum solution depends on how close \mathcal{M}' is to the CIS. When \mathcal{M}' is a superset of CIS, the modified problem should have the same solution as the optimum solution, with no loss of solution quality.

The major challenge to this approach is that the model is supposed to output a group of vectors with strict constraints: each vector needs to be an independent set, representing a valid set of non-interfering network links. This turns out to be difficult for current neural network models to directly output. The commonly used models perform well when training on “dense” data by

learning from a great number of input-output sample pairs, sufficiently close for the models to extrapolate. However, the set of scheduled links in networks are represented by sparse vectors: in a network with hundreds of potential links, typically fewer than 5% of them are scheduled at a given time. Even with many training samples, still the output space can hardly be covered to allow effective learning.

In response to such challenge, our self-supervised learning model is designed with a two-stage approach, as illustrated in Fig. 1.

The first stage is Scheduling Structure Classification (SSC), which distills the scheduling structure (represented as scheduled independent sets) of historical optimization instances into an appropriate number of scheduling classes through a properly designed clustering algorithm without additional supervision signal. The output from the SSC stage offers labeled data to train the second stage, the application identification (AID) stage, in a supervised learning manner. The SSC stage also summarizes the schedulable ISs associated with each schedule class. The AID stage is capable of predicting an optimization instance’s scheduling class given its application level information.

In the usage phase, a new problem instance is directly fed to the AID model to obtain a suggested scheduling class; the class label can then be used to retrieve the schedulable ISs learned in the SSC stage. These retrieved set of schedulable ISs (\mathcal{M}') is an approximation (predicted by the model) of the target CISs for this new instance and used in (6) to compute the approximate solution for the original optimization problem. Fig. 2 gives an illustration of the idea.

In our approach, a large number of optimization instances are solved with the traditional DCG method to serve as the training data (i.e., the historical instances). Each instance contains information about network topology, commodity source/destination deployment, commodity weights and the scheduled ISs. As the intuition of our method is to extract the mapping relationship between the application-level commodity traffic requirement and the supporting scheduling structure, we will train in a setting that statistically covers the commodity geometric distribution over the whole network. We then test the trained model in diversified commodity deployment scenarios to examine the robustness of

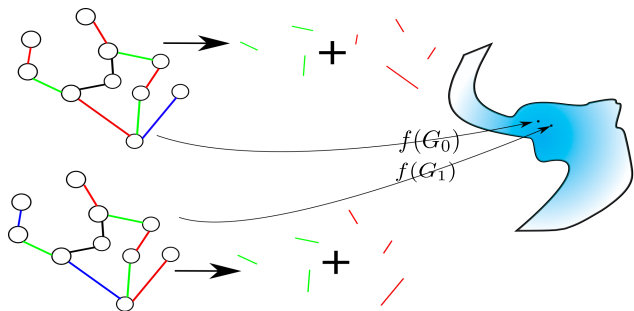


Fig. 2. Problem instances with similar scheduling are identified. G_0 and G_1 are two problem instances that are different by traffic demands and node location (not shown in the figure). They have similar scheduled ISs, and with the proposed algorithm, a mapping f is learned by the AID model so that on a manifold, these problem instances with similar link usage patterns are mapped to points close to each other. In this way, the model can help “recall” historical instances that are helpful to solving future ones.

the performance. Detailed performance evaluations are given in the numerical experiment section.

We note that the approach is not necessarily limited to the multi-hop networks. However, because the routing and scheduling in such type of networks limited by interference is much more difficult than single-hop ones, we have chosen this scenario for illustrating the power of deep learning methods.

A. Scheduling Structure Classification (SSC)

For any problem instance of this type, its solution is a time-share scheduling of a collection of CISs, which can be represented by a point in the convex hull spanned by the CIS vectors. When the network does not change significantly (e.g., the flow requirement changes or the nodes shifted in location), different optimization instances are bound to have largely overlapping CISs and differ only in the coefficients. The overlapped parts are the independent sets scheduled in both the machine learning predicted results and the conventional method’s optimum results, and the degree of such overlapping can be quantified with Jaccard similarity measure. In other words, the convex hull, determined by just one collection of CISs, can contain solutions to many other problem instances. Thus, although problem instances can look differently on the application level, they are different only if their scheduled CISs, or *scheduling structures* are different.

By this observation, it is beneficial to use historical cases’ scheduling structure for faster optimization of new cases. It is of course possible that new problem instances need to schedule different CISs, forming different convex hulls. But if we are given a sufficiently large number of historical cases, with the limited number of valid scheduling structures, we intuit that the unseen future cases can still have significant overlapping scheduling structures with known one, i.e., their convex hulls can still be close to convex hulls seen during training. This means that for two different input problem instances, their optimal scheduling could have many identical independent link subsets. The larger the overlapping between their scheduled CISs, the more similar the two problem instances are considered to be.

Such similarity in scheduling structure is not superficially observable just by routinely checking application-level problem inputs; we delegate such task of identifying similarity to a clustering algorithm. The goal is to group known instances into clusters that have similar scheduling structures, hereafter referred to as belonging to the same *scheduling class*. We take the union of the CISs from all the instances falling in the same class, denoted as UCISs, as the approximate critical ISs suggested for instances in that class. The UCISs for all the scheduling classes are stored as distilled summary of all historical experiences.

The implementation details of our clustering algorithms are as follows.

1) *Instance Representation*: The information from each problem instance can be seen as two parts: one is the network’s nodes, links, their capacities and demands, $(\mathcal{N}, \mathcal{E}, \mathcal{D})$; the other is the corresponding scheduled CIS: $\{\text{IS}_0, \dots, \text{IS}_{k-1}\}$, where k is the number of independent sets in CIS and may be different from case to case. Each of the IS is represented by an $|\mathcal{E}|$ -dimensional vector, with $|\mathcal{E}|$ as the number of links. The j -th element of the vector represents whether link j is activated in this IS: 1 if activated and 0 otherwise.

For clustering, we only use the scheduled CIS as the basis for clustering without involving the network related information. The issue is that the problem instances are likely to be scheduled with CISs of varying sizes; some instances may have CIS twice as large as others. To obtain a single vector representation, we use a pooling operation. Pooling refers to the process of “summarizing” the independent sets information for one problem instance. It would take as input several independent set vectors and output a single vector containing information from them. We require a fixed sized vector representation for ease of processing. We use averaging for the pooling operation: an instance’s scheduling structure is calculated as the mean of all these k CIS: $\text{Repr} = \frac{1}{k} \sum_{i=0}^{k-1} \text{IS}_i$. Other ways of pooling is also possible, for example, taking the summation over those ISs; we find through our experiments that the mean-based representation robustly performs well.

2) *Dimension Reduction*: With the $|\mathcal{E}|$ -dimensional vector obtained for each problem instance, clustering may proceed based on this alone. However, the dimension of these vectors increases rapidly with the size of the network. Starting from a few dozen when there are tens of nodes, their dimension can go into the range of thousands even when the number of nodes is at a modest 50.

Such high dimensionality is problematic because it is well-known that clustering algorithms converge slowly with high-dimensional data. An even more negative aspect is that the *curse of dimensionality* can seriously impact the clustering performance, because in the high-dimensional regime, the data points are often equally close to each other, thus rendering the distance-based clustering method ineffective [19].

Therefore, before the clustering, we use the idea of principal component analysis (PCA) [20] for dimension reduction. It has been applied in wireless network problems, including PCA-guided routing algorithm [21] and anomaly detection in wireless sensor networks [22], as it transforms interrelated features in a dataset into a number of uncorrelated features called principal

Algorithm 1. Dimension Reduction of the Scheduling Structure.

Input: Original data matrix $T \in \mathbb{R}^{N \times d_o}$, EVR threshold γ

Output: Reduced-dimension data Z

```

/* SVD decomposition of data;  $U, W^H$  are
   singular vector matrices, and  $S$  is the
   vector of singular values. */
 $U, S, W^H = \text{SVD}(T)$ ;
/* variations along each dimension. Squaring is
   element-wise. */
 $V' = S^2 / (n - 1)$ ;
 $B = \text{SUM}(V')$ ;
/* Sort in descending order each dimension's
   variance, and save the sorted indices in  $S$ 
   */
 $V, S = \text{Sort}(V'/B)$ ;
/* Starting from the dimension with greatest
   variation, loop until the cumulative EVR is
   greater than the threshold. */
 $t, q = 0$ ;
for  $i = 0$  to  $n - 1$  do
     $t += V[i]$ ;
     $q = q + 1$ ;
    if  $t \geq \gamma$  then break;
end
/* Retrieve the columns corresponding to the
   dimensions with greatest variance */
return  $Z = U[S[0:q]]$ 

```

components (PCs). This in effect transforms the vector representation of each instance into a vector in a smaller subspace such that the resulting smaller-sized vectors still preserve the variance in the original data. This counters the curse of dimensionality because it preserves the difference among the instance vectors and discards the dimensions with little variation, and as a result, distances between the vectors would be more salient for the clustering step to capture.

The steps of dimension reduction is listed in Algorithm 1. It takes as input the data matrix $T \in \mathbb{R}^{N \times d_o}$, whose rows are the original instance vectors, and $\gamma \in (0, 1]$, a tunable parameter representing the threshold for output principal component selection. Introducing this parameter is necessary because choosing an appropriate number of PCs is not straightforward, as it impacts the optimality of clustering. Keeping fewer PCs (e.g 2D or 3D) causes loss of critical information in the data, and retaining very large PCs causes clustering complexity using the K-means algorithm.

To solve this, we use the explained variance ratio (EVR) [23] of each dimension over all the instance vectors, and take the largest of them to the point that their combined EVR surpasses a certain threshold. We select the dimensions starting from the largest EVR and maintain the minimum number of dimensions whose cumulative EVR exceeds γ , taken to be 50% in the experiments unless otherwise noted.

Algorithm 2. Instance Clustering.

Input: Reduced-dim data $Z \in \mathbb{R}^{N \times d_o}$, number of clusters K , number of max iterations N_I , distance function $D(p_0, p_1)$

Output: Cluster labels $Y \in \mathbb{R}^N$

Uniformly randomly select one instance vector $c_1 \in Z$;

```

/* probabilities are scaled according to the
   distance with the initial point. */
Randomly select  $K - 1$  points  $c_2, \dots, c_K$  from the
rest, with the probability of a point  $z_i \in Z \setminus z_0$  being
chosen defined as  $D(z_i, z_0) / \sum_i D(z_i, z_0)$ ;
for  $i = 1$  to  $N_I$  do
    /* For each point  $z'$ , find its own cluster
       */
    for  $j = 1$  to  $N$  do
         $Y[j] = \arg \min_k D(z_j, c_k)$ ;
    end
    /* Recalculate the cluster centers */
    for  $j = 1$  to  $K$  do
         $c_j = \text{AVERAGE}(Z_j)$ ; //  $Z_j$  is the set of
        points in cluster  $j$ 
    end
end
return  $Y$ 

```

3) *Clustering Into Scheduling Classes:* After dimension reduction, the instance vectors are sufficiently far apart of each other so that clustering can be performed more easily. We use K-means++ algorithm [24] to cluster the known case data into several structural classes. This choice is corroborated in our experiments, and a comparison with the standard K-means is presented in Section IV-C.

The basic idea is that given the reduced-dimension instance vectors in Z , the aim is to choose K vectors c_1, \dots, c_K acting as centers, so as to minimize the total squared distance between each point and its closest center point:

$$\underset{c_1, \dots, c_K \in Z}{\text{minimize}} \sum_{k=1}^K \sum_{z \in Z_k} \|z - c_k\|^2,$$

where Z_k is the set of instance vectors that are closest to center point c_k more than other instance vectors. In this way, each point is assigned one of the K clusters, and in this problem, it represents the structural class it belongs to.

This problem is NP-hard if it is to be solved exactly, but in typical usage scenarios, an iterative heuristic update process is used to approximate the solution. Our clustering mechanism is thus, based on it, as shown in Algorithm 2.

Such an algorithm is chosen for the following reasons. First is that it has a low runtime: for each iteration it only costs $O(KN)$, where K is the number of clusters and N is the number of total instances. The other reason is that when updating, only the distance between each case to its cluster center is needed. This suits the network problems, because most of the scheduling

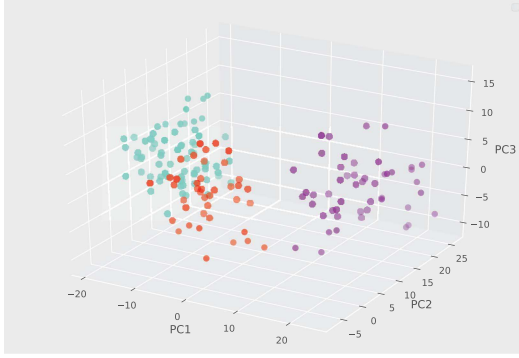


Fig. 3. Illustration of the clustering of scheduling structure.

patterns, in the reduced dimensions, are quite disparate and they naturally form small clusters. Hence it is not necessary to compare each case against all others; only the closest ones are necessary.

In our context, we do not have the prior knowledge of what the proper number of clusters to classify the scheduling structure should be. Choosing the number of classes in the SSC stage follows a heuristic process: we conducted experiments with different values of K and choose the best one. There are existing methods such as the *elbow method* [25] to find a reasonable number; but because the final performance metrics we are interested in are not easily determinable at this stage as a single number, we instead try out several candidate values before settling on a specific number. For the effects of number of classes on the performance figures, there is more discussion in Section IV-C.

In Fig. 3, we illustrate the clustering of scheduling structure over the historical instances from a 30-node network, the topology of which is given in the numerical experiments section. In order to give a clear visualization, we project all the vectors to the three dimensions associated with the top-3 dominant PCA components, and just display three scheduling classes from the clusters. Note that all the training instances will be attached with an appropriate class label based on the clustering results, and serve as labelled data to train the AID model.

B. Application Identification

The AID model is to be trained with labelled instances obtained by the SSC stage. We define a network incidence matrix $M \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ to encode the application-level information required for optimization, including the network topology, link capacities, and commodity flow source/destination deployment. Its element M_{ij} is equal to the link's capacity if there exists one from node i to node j , and 0 otherwise. The commodity flow demand nodes are represented on its diagonal elements: $M_{\text{src}_i, \text{src}_i}$ is set to $-i$ and the corresponding $M_{\text{dst}_i, \text{dst}_i}$ is set to i . Since the network nodes are indexed as $n \in \{0, 1, \dots, N-1\}$, and each element in the matrix (except diagonal ones) equal 0 or the capacity of the link, it will be conflicting to represent the source and destination nodes of multi-commodity flows with 0's and 1's, instead we represent each $M_{\text{src}, \text{src}}$ and $M_{\text{dst}, \text{dst}}$ in the matrix as $-i$ and i , where $i \in \{1, \dots, D\}$ and D is the number of commodity flow deployments.

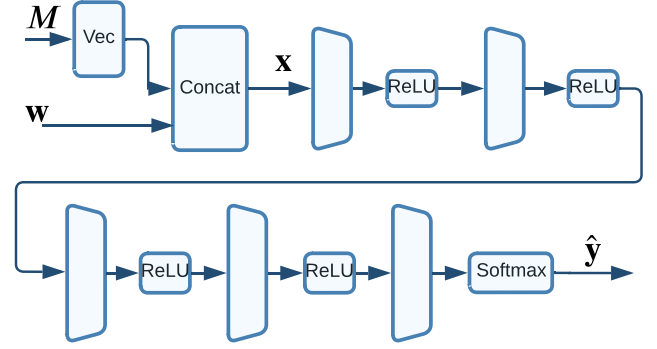


Fig. 4. The data flow of AID model.

The weights of these commodity flows, $\mathbf{w} = (w_0, \dots, w_{D_i-1})$ with D_i denoting the number of commodity flows in instance i , will also be part of the input. In sum, the problem instance is represented as a vector \mathbf{x} obtained by

$$\mathbf{x} = \text{Concat}(\text{Vec}(M), \mathbf{w}), \quad (7)$$

where *Concat* is a vector concatenation operation and *Vec* converts a 2-D matrix into a 1-D vector in the row-major order.

The input will pass through several fully-connected layers, and the final layer is to output the estimated probability distribution for each scheduling class, as shown in the Fig. 4. With the softmax activation function, the output $\hat{\mathbf{y}}$ can be summarized as follows:

$$\text{FC}_i(\mathbf{x}) \triangleq \text{ReLU}(\mathbf{W}_i \mathbf{x} + \mathbf{b}_i) \quad (8)$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_5(\text{FC}_4(\text{FC}_3(\text{FC}_2(\text{FC}_1(\mathbf{x})))))) + \mathbf{b}_5) \quad (9)$$

During inference, the class with the maximum predicted probability is used. The cross-entropy loss is used for minimizing the distance between predicted and actual class label distribution. Note that because the dimensions of the neural network parameters correspond to the network dimensions, this means that our solution assumes that no new nodes should join the network, or any existing nodes should leave the network.

IV. NUMERICAL EXPERIMENTS

In this section, we present the numerical results to evaluate the performance of the proposed self-supervised learning based optimization. Our method will be applied to solve optimization instances with different number of commodity flows and different commodity weights over the same topology.

A. Experiments Settings

We consider a network formed by nodes randomly distributed within a fixed 1 Km by 1 Km square area with a minimum distance of 30 m. The communication range and interference range are set to be the same for all nodes, at 50 m and 70 m, respectively. Each link capacity is calculated based on the Shannon formula using a preset signal power 17 dBm and noise power -127 dBm. We specifically study two network topology, one medium-sized with 30 nodes and one large-sized with 100

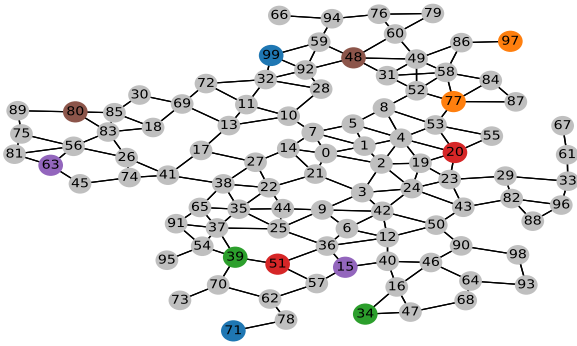


Fig. 5. A 100-node network topology with 6 commodity flows.

nodes. The large 100-node topology is shown in Fig. 5, with illustration of a deployment instance of 6 multi-hop commodity flows; the source/destination nodes of different commodities are differentiated with colors.

The training data along with the scheduled CIS are computed with DCG algorithm, under different settings of 1 to 6 commodity flow requirements. However, the testing data consist of larger commodity flow deployments ranging from 1 flow to 10 flows.

The source and destination nodes are randomly selected and commodity weights are changed crossing different instances. The consideration is to give a good geometric coverage to the whole network in the statistical sense. The trained model is tested in diverse commodity deployment scenarios to examine the robustness of the performance. The dataset contains over 100,000 problem instances and 10% of them are used as dedicated testing set. Even though this is not a trivial number, it is possible by using multiple servers running over a period of time. Also note that the DCG algorithm is only executed during the data generation and not in the training, so the training itself is not slowed down by the use of DCG. We note that it is possible to add additional data after the training is complete by following the same procedures in clustering and label generation, but there is the limitation that the number of classes cannot dynamically scale with more data. The experiments are run on Google Colab, equipped with Nvidia K80 GPUs and Intel Xeon processors @ 2.3 GHz. Python software packages Numpy [26] and Tensorflow [27] are used. The configurations of hyperparameters for training the AID model are listed in Table III.

For all experiments under either medium- or large-sized network setting, we aim to quantify the solution time reduction and the loss in the solution quality by using the proposed method. Towards this end, the following measures are defined:

$$AR = \frac{SOL_{ML}}{SOL_{DCG}} \quad CTR = \frac{t_{DCG} - t_{ML}}{t_{DCG}} \quad (10)$$

a) *Approximation ratio (AR)*: defined as the ratio between the optimal network flow computed using the proposed approach, denoted as SOL_{ML} , and that computed using the conventional DCG algorithm, denoted as SOL_{DCG} . Note that this definition is different from the conventional use of the term because the solutions from DCG algorithm are not always guaranteed to the optimum. Nevertheless, the solutions of DCG are high quality

enough that we consider them to be sufficient for benchmark purposes.

Computation time reduction (CTR): the ratio between the computation time reduction (benefiting from the proposed method), indicated as $t_{DCG} - t_{ML}$, and that by the conventional method denoted as t_{DCG} . In calculating the former, we consider the total computation time, which accounts for both the AID model's inference time and the LP computation time with the suggested UCISs.

B. Performance Evaluation in a Medium-Sized Network

We apply the proposed method to solve the testing instances within the prepared dataset, and record the computation time and the optimum flow for each case. To demonstrate the robustness of our method, we test in different commodity flow deployment scenarios. The results for the 30-node network are reported in Table II.

As shown from the CTR and AR figures, we can see that our method achieves a significant time reduction consistently across different settings. Across the different number of flows, the approximation ratio remains over 84%, meaning that the optimum values are almost the same to the original solution, with computation time reduction at least 77%. Such robust performance is because our method can learn and exploit the internal scheduling structure.

From Table II, in the scenarios of high traffic flow regime, we do observe slight degrading in the AR performance (from 98% to 84%). This is expected given that the problem gets more difficult and the solution structures become richer as the number of flows increases, and it is not avoidable to have better solutions not covered by the model's capability.

C. Effects of Clustering Algorithms and the Number of Structural Classes

In this subsection, we document the performance differences under the influence of three design choices: the used clustering algorithm, the number of structural classes, and the effects of vector dimension reduction. We conducted the experiments on problem instances with different traffic demands, collected the performance metrics data and plotted the results in Figs. 6 and 7.

First, we can observe that using the dimension reduction technique results in an observable time benefit, as shown in Fig. 7 where the PCA-enabled curves are generally higher than those that are not, suggesting that on average this is indeed faster in the solution process. And this benefit is achieved at a small price, demonstrated in Fig. 6 that the approximation ratio remained over 96% at low-flow demand regime. In high flow demand scenarios, however, the PCA-enabled method leads to even higher approximation ratio, showing that the benefit goes beyond saving the time for processing long vectors, and can lead to a more representative vector for the structural information in the data.

A main reason of favoring K-means++ over the standard K-means is that its initialization scheme could make the resulting clustering process easier and the clustering may be of

TABLE II

THE APPROXIMATION RATIO (AR) AND THE COMPUTATION TIME REDUCTION RATIO (CTR) COMPARISON FOR DIFFERENT COMMODITY FLOW DEPLOYMENT SCENARIOS IN THE 30-NODES NETWORK USING KMEANS++ WITH 500 CLASSES. BOTH PERFORMANCE METRICS ARE THE LARGER, THE BETTER. COMPUTATION TIME IS IN THE UNIT OF SECONDS. THE OPTIMAL FLOW IS IN THE UNIT OF MBITS/S

No. of commodity flows	1	2	3	4	5	6
t_{DCG}	1.97 ± 0.64	2.49 ± 0.54	2.61 ± 0.47	2.90 ± 0.47	2.99 ± 0.39	3.10 ± 0.42
t_{ML}	0.45 ± 0.15	0.53 ± 0.20	0.62 ± 0.30	0.65 ± 0.34	0.68 ± 0.40	0.62 ± 0.35
CTR	0.77	0.79	0.76	0.78	0.77	0.80
SOL_{DCG}	3.47 ± 1.14	5.52 ± 1.64	7.10 ± 2.16	8.24 ± 2.63	9.24 ± 2.92	9.99 ± 3.04
SOL_{ML}	3.46 ± 1.14	5.50 ± 1.64	7.00 ± 2.17	8.11 ± 2.65	9.05 ± 2.93	9.71 ± 3.07
AR	1.00	1.00	0.99	0.98	0.98	0.97

No. of commodity flows	7	8	9	10
t_{DCG}	8.00 ± 3.87	9.51 ± 3.82	10.11 ± 3.94	10.68 ± 4.06
t_{ML}	0.41 ± 0.12	0.39 ± 0.11	0.39 ± 0.11	0.43 ± 0.12
CTR	0.95	0.96	0.96	0.96
SOL_{DCG}	10.96 ± 3.30	12.04 ± 3.56	12.48 ± 3.57	12.92 ± 3.65
SOL_{ML}	9.41 ± 3.09	10.19 ± 3.19	10.60 ± 3.27	10.90 ± 3.34
AR	0.86	0.85	0.85	0.84

TABLE III

LIST OF PARAMETERS AND HYPERPARAMETERS FOR TRAINING THE AID MODEL

Name	Value
Epochs	500
Batch size	50
Layers	5
Optimizer	Adam
Regularization	Early stopping
W_1	$64 \times 906, 64 \times 10006$
W_2, W_3	64×64
W_4, W_5	$128 \times 64, 250 \times 128$

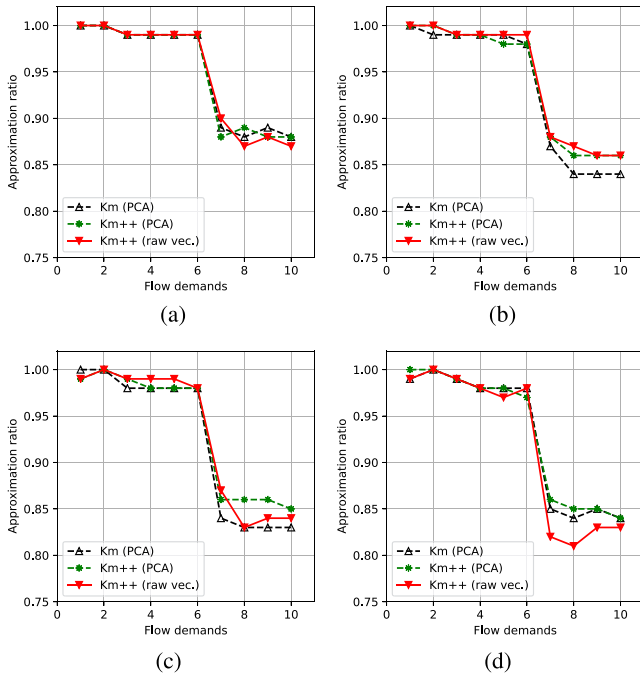


Fig. 6. Approximation ratios for problem instances with different number of structural classes and number of flow demands. The number is the higher the better. (a) 200 classes, (b) 300 classes, (c) 400 classes, (d) 500 classes.

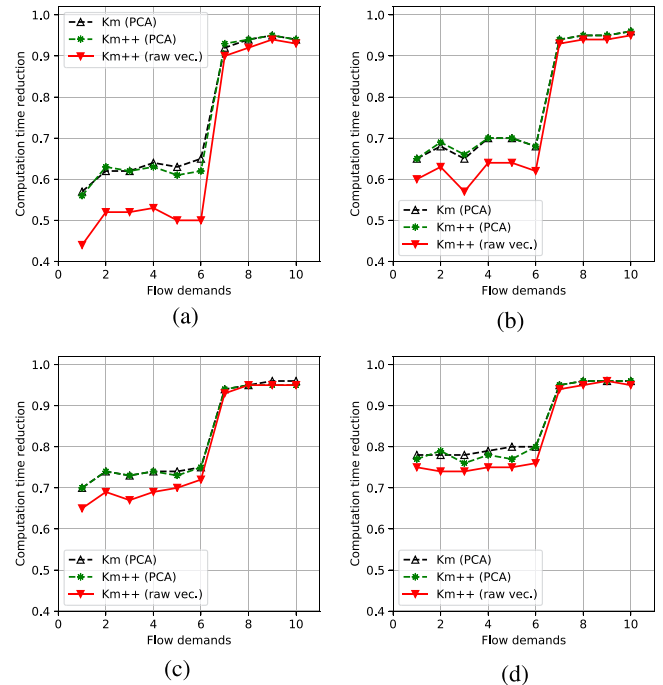


Fig. 7. Computation time reductions for problem instances with different network sizes the number of flow demands. The number is the higher the better. (a) 200 classes, (b) 300 classes, (c) 400 classes, (d) 500 classes.

a higher quality. This is shown by the higher approximation ratios achieved by K-means++ in Fig. 6, especially when the number of traffic demands is high. And this benefit is gained with very little loss in the computation time, evidenced by the almost overlapping dotted green and black curves in Fig. 7.

The effect of the number of structural classes to be used is clearly demonstrated in these plots. We observe that as for the approximation ratio, there is no indication that more classes are better: in the high flow demand regime (greater than 6 traffic

TABLE IV
 PERFORMANCE COMPARISON OF OBCR AND PROPOSED SOLUTION

No. of commodity flows	1	2	3	4	5	6
t_{OBCR}	3.89 ± 0.11	3.89 ± 0.14	3.93 ± 0.15	4.20 ± 0.21	4.25 ± 0.24	4.30 ± 0.18
t_{ML}	0.49 ± 0.25	1.58 ± 0.82	1.95 ± 1.58	2.10 ± 0.84	2.28 ± 0.88	2.29 ± 0.87
SOL_{OBCR}	3.09 ± 0.89	4.32 ± 1.36	5.22 ± 1.36	6.12 ± 1.56	6.92 ± 1.56	7.49 ± 1.88
SOL_{ML}	3.94 ± 1.05	6.09 ± 1.53	7.83 ± 1.78	9.31 ± 2.06	10.40 ± 2.20	11.16 ± 2.55

No. of commodity flows	7	8	9	10
t_{OBCR}	4.23 ± 0.82	4.05 ± 0.86	4.10 ± 0.82	4.16 ± 0.86
t_{ML}	2.37 ± 0.12	2.48 ± 0.11	2.46 ± 0.11	2.42 ± 0.12
SOL_{OBCR}	8.06 ± 1.89	8.70 ± 2.07	8.86 ± 1.91	9.54 ± 2.05
SOL_{ML}	9.75 ± 2.32	10.47 ± 2.52	10.66 ± 2.36	11.41 ± 2.65

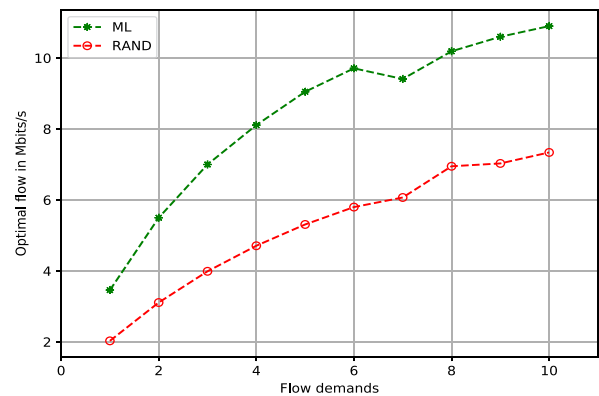
demand pairs), the average approximation ratio has an about 5% decrease between the instances predicted on 200- and 500 classes. This can be explained by the fact that with the larger number of classes, each class now have less information from the related instances, so choosing a single class can lead to a less informed start. However, for reducing the computation time, a higher number of structural classes is beneficial. An improvement as high as 10% in computation time reduction can be achieved for the lower traffic demand regime. This is also expected because with more classes, each structural class is smaller and it means less time in processing high dimensional calculations.

D. A Testimony to ML Intelligence

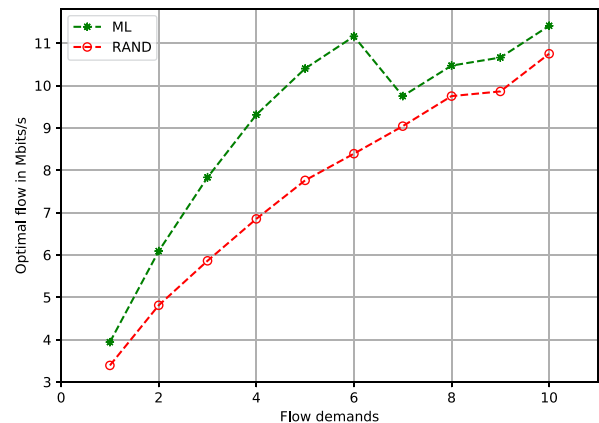
To further test that ML model in our approach indeed extracts valuable knowledge that could not be available otherwise, we use a fast, simple heuristic method as a baseline for the performance comparison. We know that our proposed method suggests a UCIS with a certain average size to a new instance, which will be directly used for one-round LP to get the high-quality approximate results with much reduced computation time, compared to the traditional DCG. To get evidence that such UCIS recommendation is not just good by chance, we randomly search the same amount of ISs and use them as the CISs to test the performance.

The performance under the random search method is reported in Fig. 8 indicated as the red curve. As a testimony of ML intelligence, SOL_{Rand} is significantly less than the ML performance SOL_{ML} in all the scenarios. This shows that the the ML model predicts the ISs that are highly relevant to the current problem, and good performance is very unlikely to be gained from chance.

Next, we compare our method with another sophisticated heuristic algorithm (hereafter called OBCR) for generating maximal independent set [28]. This method uses a randomized approach to achieve distributed execution with provably efficient message exchange, making it a practical algorithm that can be used in a networking setting for obtaining independent sets. For each instance, we use this algorithm to obtain as many as maximal independent sets as possible under a certain time budget, and then use them for the LP solver to generate a solution with one round of linear programming. This is the same process



(a)

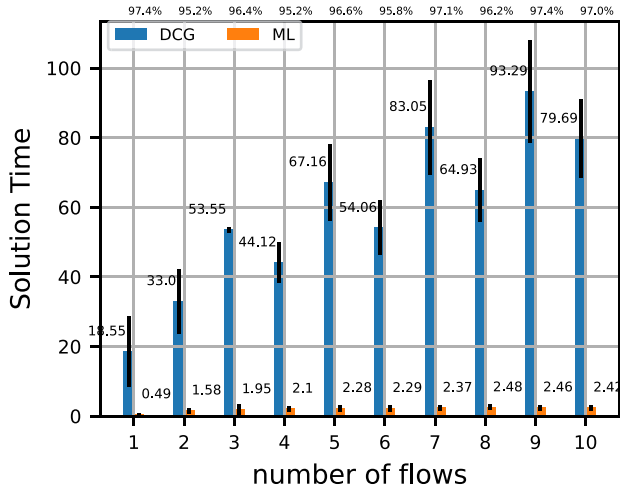


(b)

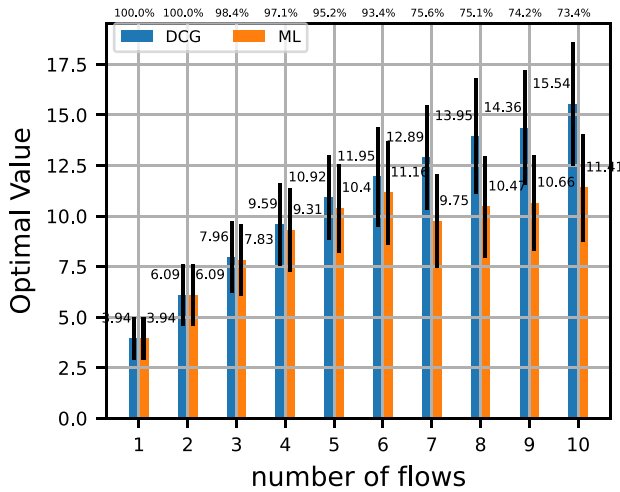
Fig. 8. Comparison of proposed method to random IS. (a) 30-nodes network, (b) 100-nodes network.

as our ML-based approach, with the only difference that the UCIS are purely generated from OBCR instead of machine learning steps.

We perform the experiments with samples from 100-node networks with different flow demands, and allow OBCR twice the execution time of our method's, to allow any inefficiency in the implementation. We list the computation time and solution quality in Table IV. Again, the solution quality of OBCR is noticeably lower than that of our method even with more time



(a)



(b)

Fig. 9. Performance in the 100-nodes network. On the horizontal axis, we group the performance by scenarios according to the number of deployed commodity flows. Performance figures for the proposed method and the DCG method are marked on the plot. The AR and CTR values are also shown.

budget. This confirms that the patterns captured by the neural network model are indeed more powerful and subtle than what can be described by designer's intuitive heuristic.

E. Performance in a Large-Sized Network.

We also conduct the testing in a large-sized network with 100 nodes. The results are plotted in Fig. 9 for illustration. The computed optimal flow and the computation time are marked on top of the bar and the AR and CTR values are listed at the top of the plot. All the positive performance evaluation statements obtained in the 30-node network also apply to this large-sized network. When we compare the performance between the 100-nodes network and the 30-nodes one, under a similar setting with the same number of commodity deployment, we notice that the CTR in the large-sized network is moderately better than that from the smaller network; it is due to the fact that searching for

ISs in the conventional DCG does takes up a greater portion of solution time in a larger network, but is avoided with the our ML based method, reducing the per-case time from minutes to seconds. The AR is however slightly degraded in the larger-network case, it is because the larger network has more complex scheduling structure to deal with.

F. Performance Evaluation on Max-Min Type Flows

In this section, we consider the effects of our solution for a related but different type of problem. Instead of finding the maximum of the weighted sum of all commodity flows, this problem seeks to maximize the minimum of flows. In terms of the notations used in (5), the problem is turned into

$$\text{maximize } \min_d r_d$$

$$\{f_d(u,v)\}, \{\alpha_m\}$$

s.t. constraints(2), (3), (4),

$$f_d(u,v) \geq 0, \quad \forall (u,v) \in \mathcal{L}, \forall d$$

$$\alpha_m \geq 0, \quad m \in \mathcal{M}. \quad (11)$$

This problem is still a linear programming problem, because the minimization in the objective can be turned into a linear constraint by introducing an auxiliary variable r' :

$$\text{maximize } r'$$

$$\{f_d(u,v)\}, \{\alpha_m\}$$

s.t. constraints (2), (3), (4),

$$r' \leq r_d, \quad \forall d$$

$$f_d(u,v) \geq 0, \quad \forall (u,v) \in \mathcal{L}, \forall d$$

$$\alpha_m \geq 0, \quad m \in \mathcal{M}. \quad (12)$$

We choose this because such a scenario is practical in flow optimization. Rather than maximizing the weighted sum of all types of flows, this is a fairness-oriented system goal, focusing on improving the lower-bound of all the commodity flows. Such difference induces a different type of solution space and its structure, and can provide a good viewing angle for how the algorithm is generally applicable.

To start, we generate the training dataset separately from the other parts by using this new problem setting. For simplicity, instances featuring 100-node network sizes and 2-4 demands are featured. Other than the optimization objective, all the parameters for generation are not altered. During the training, the instance representation does not contain commodity weights as they are not needed. The results are shown in Fig. 10, and we can see a similar pattern that the solution time is reduced greatly at only a fraction of the solution quality loss.

The choice of this example here is by no means due to that our method only applies to cases with different optimization objectives, but more of a practical concern: because of the limit of the paper's scope, it would be confusing to introduce an entirely new type of problem. In fact, our method can be applied in a wide variety of applications, including integrated access and backhaul and wireless sensor networks, because our model makes very generic assumptions of the network and that the solution is entirely data-driven.

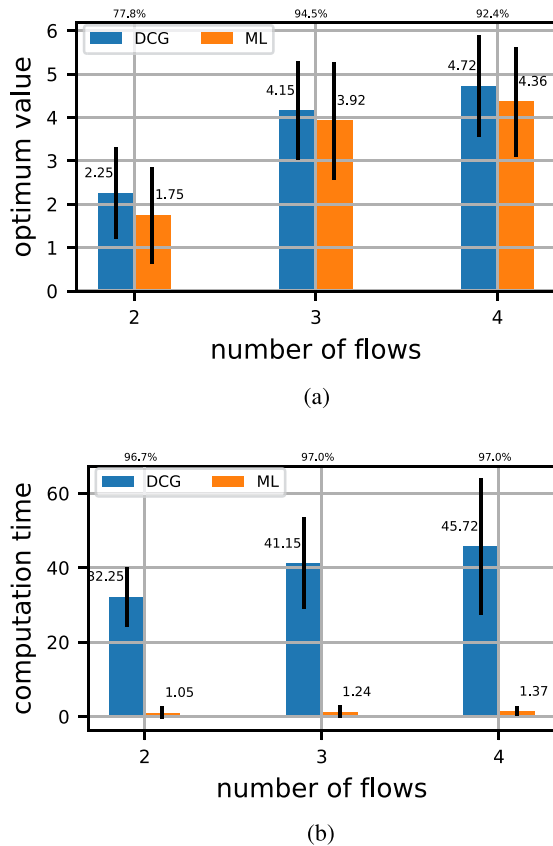


Fig. 10. Performance in the 100-nodes network optimizing the max-min type of flows. On the horizontal axis, we group the performance by scenarios according to the number of deployed commodity flows. Performance figures for the proposed method and the DCG method are marked on the plot. The AR and CTR values are also shown.

V. RELATED WORK

A. Conventional Wireless Network Optimization.

Wireless network optimization had been a key research area in the recent two decades. The basic methodology is to compute the resource allocation aspects such as channel assignment, base station association, scheduling, and power control using various mathematical programming algorithms [29]. Due to the complex interference relationship, wireless network optimization is NP-hard in general, and the major thread of efforts in the community is the development of various approximation algorithms [30], [31]. The studies had also been extended from single-radio single-channel context to complex multi-radio multi-channel context [32], [33], [34]. A particular issue inspiring the machine learning study in [6] and this paper is that a new optimization problem instance is always solved either from scratch or with a trivial re-optimization approach [35]; machine learning aims to exploit the historical computation effort to benefit new optimization instances.

B. Neural Network Approaches for Combinatorial Problems.

Deep learning-based methods are used to efficiently solve combinatorial optimization problems. The major line of research is to use sequential modeling and graph neural networks to obtain

end-to-end solutions. Pointer network [36], a model based on multi-headed attention mechanism, solves variable-sized combinatorial problems by using the attention scores as selection criteria, and this approach is shown to achieve reasonable performance level with classic problems including Traveling Salesman Problem and Delaunay triangulation. This idea is further expanded to solve a vehicle routing problem [37]. The basic pattern of an encoder-decoder setup has found applications for solving problems under similar circumstances [38], [39], [40].

C. Machine Learning in Wireless Networking

In recent years, there have been studies on developing machine-learning based approaches to tackle wireless networking related problems. The existing studies can be roughly categorized into two themes, depending on the angle to exploit the advantages of machine learning. One theme is to leverage the capability of machine learning to identify the complex mapping relationships among a large number of parameters, essentially treating the learning model as a black-box approximator. These tend to focus on a purely data-driven end-to-end approach [5], [41], [42]. Specially, the work [43] is applied in traffic prediction for proactive resource allocation, and [44] uses deep learning for executing network functions. Another theme is to adopt the deep reinforcement learning (DRL) technique for on-line control problems. Indeed, many resource allocation problems in wireless networking can be cast as a Markovian decision process and can significantly benefit from the DRL techniques [4].

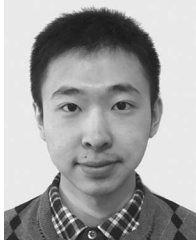
While the capacity optimization or the optimal resource allocation in the context of multi-hop wireless networks had been a key research area in recent two decades [30], [31], the state of arts was limited to the development of approximation algorithms and had limited recent progress [45], [46]. ML related studies on multi-hop wireless networking are also very limited. The work in [6] proposed the idea to leverage historic computing experiences by ML to facilitate multi-hop wireless network flow optimization. However, the ML implementation there was limited to trimming unimportant links off the network topology to reduce the problem scale, instead of contributing a new optimization method. The work [47] was recently enhanced with the graph embedding technique to make the trimming technique applicable to different topologies. However, these papers make use of the past experience in an indirect way: they do not improve the solution process through a partial reuse of the historical solutions, which is the angle taken of this work that is not yet reported in the related literature, to the best of the authors' knowledge.

VI. CONCLUSION

In this paper, we present a self-supervised learning approach to accelerate optimization in multi-hop wireless networks. Extensive numerical results have shown significant gain in the trade-off between computation overhead and the solution quality. Our study demonstrates that machine learning has great potential in enabling disruptive methods to address certain classic NP-hard problems. It is expected that our study can attract more attention into this exciting direction of research.

REFERENCES

- [1] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proc. IEEE 24th Annu. Joint Conf. IEEE Comput. Commun. Societies*, vol. 4, 2005, pp. 2492–2501.
- [2] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number," in *Proc. 38th Annu. ACM Symp. Theory Comput.*, 2006, pp. 681–690.
- [3] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. 43rd IEEE Conf. Decis. Control*, 2004, vol. 2, pp. 1484–1489.
- [4] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. Conf. ACM special Int. Group Data Commun.*, 2018, pp. 191–205.
- [5] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1248–1261, Jun. 2019.
- [6] L. Liu, B. Yin, S. Zhang, X. Cao, and Y. Cheng, "Deep learning meets wireless network optimization: Identify critical links," *IEEE Trans. Neww. Sci. Eng.*, vol. 7, no. 1, pp. 167–180, Jan./Mar. 2020.
- [7] Y. Cheng, B. Yin, and S. Zhang, "Deep learning for wireless networking: The next frontier," *IEEE Wireless Commun.*, vol. 28, no. 6, pp. 176–183, Dec. 2021.
- [8] S. Basagni, "Finding a maximal weighted independent set in wireless networks," *Telecommun. Syst.*, vol. 18, no. 1, pp. 155–168, 2001.
- [9] M. Cao, X. Wang, S.-J. Kim, and M. Madhian, "Multi-hop wireless backhaul networks: A cross-layer design paradigm," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 4, pp. 738–748, May 2007.
- [10] M. E. Rasekh, D. Guo, and U. Madhow, "Joint routing and resource allocation for millimeter wave picocellular backhaul," *IEEE Trans. Wireless Commun.*, vol. 19, no. 2, pp. 783–794, Feb. 2020.
- [11] M. A. Al-Jarrah, E. Alsusa, A. Al-Dweik, and M.-S. Alouini, "Performance analysis of wireless mesh backhauling using intelligent reflecting surfaces," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3597–3610, Jun. 2021.
- [12] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 2714–2741, Oct./Dec. 2018.
- [13] D. Liu, J. Zhang, J. Cui, S.-X. Ng, R. G. Maunder, and L. Hanzo, "Deep learning aided routing for space-air-ground integrated networks relying on real satellite, flight, and shipping data," *IEEE Wireless Commun.*, vol. 29, no. 2, pp. 177–184, Apr. 2022.
- [14] Y. Liu, A. Tang, and X. Wang, "Joint incentive and resource allocation design for user provided network under 5G integrated access and backhaul networks," *IEEE Trans. Neww. Sci. Eng.*, vol. 7, no. 2, pp. 673–685, Apr./Jun. 2020.
- [15] M. Pagin, T. Zugno, M. Polese, and M. Zorzi, "Resource management for 5G NR integrated access and backhaul: A semi-centralized approach," *IEEE Trans. Wireless Commun.*, vol. 21, no. 2, pp. 753–767, Feb. 2022.
- [16] L. R. Ford Jr. and D. R. Fulkerson, "A suggested computation for maximal multi-commodity network flows," *Manage. Sci.*, vol. 5, no. 1, pp. 97–101, 1958.
- [17] 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. 15th ACM Int. Symp. Mobile ad hoc Netw. Comput.*, 2014, pp. 23–32.
- [18] 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.
- [19] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. from Data*, vol. 3, no. 1, pp. 1–58, 2009.
- [20] R. Bro and A. K. Smilde, "Principal component analysis," *Anal. Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.
- [21] G. Chen, L. Tan, Y. Gong, and W. Zhang, "PCA-guided routing algorithm for wireless sensor networks," *J. Comput. Netw. Commun.*, vol. 2012, 2012, Art. no. 427246.
- [22] D. Brauckhoff, K. Salamatian, and M. May, "Applying PCA for traffic anomaly detection: Problems and solutions," in *Proc. IEEE : Int. Conf. Comput. Commun.*, 2009, pp. 2866–2870.
- [23] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. Lanckriet, "A direct formulation for sparse PCA using semidefinite programming," *SIAM Rev.*, vol. 49, no. 3, pp. 434–448, 2007.
- [24] D. Arthur, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.
- [25] F. Liu and Y. Deng, "Determine the number of unknown targets in open world based on elbow method," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 5, pp. 986–995, May 2021.
- [26] C. R. Harris et al., "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
- [27] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.
- [28] Y. Métivier, J. M. Robson, N. Saheb-Djahromi, and A. Zemhari, "An optimal bit complexity randomized distributed mis algorithm," *Distrib. Comput.*, vol. 23, no. 5, pp. 331–340, 2011.
- [29] Z. Han and K. R. Liu, *Resource Allocation for Wireless Networks: Basics, Techniques, and Applications*. Cambridge, MA, USA: Cambridge Univ. Press, 2008.
- [30] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Netw.*, vol. 11, no. 4, pp. 471–487, 2005.
- [31] L. Georgiadis et al., "Resource allocation and cross-layer control in wireless networks," *Foundations Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
- [32] V. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, "Algorithmic aspects of capacity in wireless networks," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2005, pp. 133–144.
- [33] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 133–144, 2004.
- [34] L. Liu, Y. Cheng, X. Cao, S. Zhou, Z. Niu, and P. Wang, "Joint optimization of scheduling and power control in wireless networks: Multi-dimensional modeling and decomposition," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1585–1600, Jul. 2019.
- [35] D. Bertsekas, *Network Optimization: Continuous and Discrete Models*, vol. 8. Nashua, NH, USA: Athena Scientific, 1998.
- [36] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, vol. 28, pp. 2692–2700.
- [37] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–25.
- [38] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 13, pp. 12042–12049.
- [39] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 537–546.
- [40] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 15580–15592.
- [41] J. Jiang, S. Sun, V. Sekar, and H. Zhang, "Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation," in *Proc. 14th USENIX Symp. Networked Syst. Des. Implementation*, 2017, pp. 393–406.
- [42] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 50–56.
- [43] J. Wang et al., "Spatiotemporal modeling and prediction in cellular networks: A Big Data enabled deep learning approach," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2017, pp. 1–9.
- [44] H. Ye, L. Liang, G. Y. Li, and B.-H. Juang, "Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3133–3143, May 2020.
- [45] P. Dutta, V. Mhatre, D. Panigrahi, and R. Rastogi, "Joint routing and scheduling in multi-hop wireless networks with directional antennas," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2010, pp. 1–5.
- [46] B. Ji, C. Joo, and N. Shroff, "Throughput-optimal scheduling in multihop wireless networks without per-flow information," *IEEE/ACM Trans. On Netw.*, vol. 21, no. 2, pp. 634–647, Apr. 2013.
- [47] S. Zhang, B. Yin, W. Zhang, and Y. Cheng, "Topology aware deep learning for wireless network optimization," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9791–9805, Nov. 2022.



Shuai Zhang (Student Member, IEEE) received the B.Eng. and M.S. degrees from Zhejiang University, Hangzhou, China, and the University of California, Los Angeles, Los Angeles, CA, USA, in 2013 and 2015, respectively, and the Ph.D. degree in computer engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 2022. His research interests include wireless communication and distributed learning.



Oluwaseun T. Ajayi (Student Member, IEEE) received the B.Sc. degree in telecommunication science from the University of Ilorin, Nigeria, in 2018. He is currently working toward the Ph.D. degree with Illinois Institute of Technology. His research interests include machine learning in wireless networks, information freshness optimization, and vehicular communications.



Yu Cheng (Senior Member, IEEE) 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 research interests include wireless network performance analysis, information freshness, machine learning, network security, and cloud computing. He was the recipient of Best Paper Award at QShine 2007, IEEE ICC 2011, and a Runner-Up Best Paper Award at ACM MobiHoc 2014, National Science Foundation (NSF) CAREER Award in 2011 and IIT Sigma Xi Research Award in the junior faculty division in 2013. He has served as several Symposium Co-Chairs for IEEE ICC and IEEE GLOBECOM, and Technical Program Committee (TPC) Co-Chair for IEEE/CIC ICC 2015, ICNC 2015, and WASA 2011. He was a founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc Distinguished Lecturer in 2016–2017. He is an Associate Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE INTERNET OF THINGS JOURNAL, and IEEE WIRELESS COMMUNICATIONS.